بسم الله الرحمن الرحيم

سبحانك لا علم لنا إلا ما علمتنا إنك أنت العليم الحكيم

المقدمة :

الحمد لله ماحمده الحامدون وغفل عن حمده الغافلون والصلاة والسلام على عبده ورسوله محمد صلاة بعدد ذرات الخلائق وما يكون . ورضاك اللهم عن آله الطيبين وصحبه المكرمين المبجلين أجمعين وبعد ،

يقدم هذا الجزء من الكتاب الخطوات الأولي لتعليم لغة الجافا سكربت ، وربما يحتاج القارئ أن يكون ملما بأساسيات لغة الهتمل html الخاصة بتكوين صفحات الويب

في حالة وجود أي أخطاء أرجو اعلامي عن الخطأ علي العنوان التالي
a_elhussein@hotmail.com

وأرجو من كل من أستفاد من هذا الكتاب أن يدعوا لي بالتوفيق في الدنيا والأخرة

إهداء :

أهدي هذا الكتاب إلي الجيل القادم الذي يعز الله به الإسلام

وقد شاء الله أن أنهي هذا الجزء من الكتاب في يوم ميلاد النبي الأمي محمد بن عبد الله صلي الله عليه وسلم وهو أول معلم و خير معلم بما علمه الله ، فأحمد الله علي أن هداني لهذا وما كنا لنهتديا لولا أن هدانا الله .

وإنا مادامت فيا الحياة باذل جهدي وعقلي ومستفرغ طاقتي في العلم وذلك لثلاثة أمور

- إفادة من يطلب العلم في حياتي وبعد مماتي
- ذخيرة لي في قبري ويوم حسابي
- رفعة لسلطان المسلمين

تأليف : الحسين محمد علي

المحتويات

الفصل الأول

مقدمة للغة الجافا سكربت

سوف نناقش إن شاء الله في هذا الفصل النقاط التالية :

- ماهي الجافا سكربت
- أصل الجافا سكربت
- مميزات الجافا سكربت
- لماذا أتعلم الجافا سكربت
- ماهو الفرق بين الجافا .. والجافا سكربت
- ما المقصود بشفرة البرنامج أو السورس كود
- متصفحي النتسكيب والإكسبلورر
- برنامجك الأول "Hello World"

ماهي الجافا سكربت

الجافا سكربت وبكل بساطه لغة من لغات البرمجة ان صح التعبير .. مهامها الأساسيه بث الحياة في صفحات الويب المكتوبة بلغة ال HTML وتعطيك امكانية التحكم بكل جزء في صفحة الوب، من ال forms الى الوصلات بل وحتى بعض الوظائف الخارجيه.

أصل الجافا سكربت

الجافا سكربت صممت من قبل Netscape لإضافة بعض الحيوية الى صفحات الويب، طبعا لايفوتني أن أذكر أن ال Java في الاصل صممت من قبل شركة Sun وهي المالك الأساسي للغة .

مميزات الجافا سكربت

الجافا سكربت تختلف عن أغلب لغات البرمجة الأخرى في كونها سهلة التحكم ، حتى للأشخاص الذين لم يكتبوا بأي لغة برمجة أخرى في حياتهم كلها. بكل بساطة،الجافا سكربت هي الهواء لصفحتك.

بالإضافة إلي أنها برمجه كائنية الإعتماد Object base اي تعتمد علي وجود بعض الفئات ان صح التعبير مبنية بداخل اللغة يمكن إستخدامها بكل سهولة مثل الكائن window و الكائن document .

توفر لغة الجافا سكربت التعامل مع الأحداث events

تعمل لغة الجافا سكربت من خلال جميع أنظمة التشغيل مثل الويندوز و اللينكس ، فهي لا تعتمد علي نظام التشغيل Platform independent .

الجافا سكربت هي case sensitive language ، أي أن الكلمة المكتوبه بالحروف اللاتينيه الصغيره تختلف عن نفس الكلمة مكتوبة بالأحرف الكبيره
مثلا:
Naser غير naser غير naSer ، وهكذا..

يفترض الى درجة الإلزام أن ينتهي كل سطر بفاصلة منقوطة:

```
var x = 3;
```

الجافا سكربت كما الهتمل HTML تتجاهل المساحات الخالية ، والسطور الجديدة ، مثلا:

```
var x=4 هي نفسها var x = 4
```

مع ملاحظة أنه لابد من ترك مسافة خالية على الأقل بعد أي مصطلح من مصطلحات الجافا

لماذا أتعلم الجافا سكربت

أول مايخطر في بالي إجابة على هذا السؤال هو : القوة .. الحريه .. الإبداع . لأنه بمجرد الكتابه بال HTML فأنت مقيد بأن تظل صفحتك كما هي ثابته لاتتغير الا بتغيير الكود ، وهذا غير مقبول بتاتا في عالم التكنولوجيا الجديده .وبواسطة الجافا سكربت يمكنك حتى الباس صفحتك حلة جديدة من ألوان الخلفيه .. وأنواع الخطوط .. وحتى الصور ..حسب التوقيت اليومي للزائر وبدون أي تدخل منك!

وشيء مهم أيضا .. الجافا سكربت تعتبر من أبسط اللغات ! الكل يستطيع تعلمها .. نعم الكل .. .ألا توافقني الرأي ، أن كتابة الكود الخاص بك مباشرة أفضل ألف مرة من البحث عنه في الشبكه .

وربما لاتجد مبتغاك بسهولة ! أو قد لاتجدة إطلاقا.

ماهو الفرق بين الجافا .. والجافا سكربت

الفرق بينهما كبير . نعم فالجافا أقوي بكثير من الجافا سكربت ، وأكثر تعقيدا ، وللأسف ، أصعب في التحكم . وهي تأتي في نفس مستوى لغتي السي ، والسي بلس بلس .

بالإضافه الى أنك تحتاج الى برامج خاصة للكتابة بلغة الجافا ، بينما الجافا سكربت أبسط بكثير ! يمكنك بمجرد فتح أي برنامج تحرير نصوص مثل النوتة NOTEPAD كتابة السكربت كاملا !!! صدق أولا تصدق . لن تحتاج الى شي آخر سوى المتصفح لترى النتيجه .

ما المقصود بشفرة البرنامج أو السورس كود

الكود أو السورس كود : هو مجموعة الأوامر التي تكتب مجتمعة أو متفرقة ليعمل البرنامج بصورتة الرئيسيه ، أي البنية التحتية للبرنامج ، وتكون دائما مكتوبه كتابة ويمكن عرضها بأي برنامج تحرير نصوص عادي مثل ال NOTEPAD .

متصفحي النتسكيب والإكسبلورر

هل بإمكان الكود الذي كتبته بالجافا سكربت العمل على متصفحي النتسكيب والإكسبلورر بدون أية مشاكل ؟

للأسف ، الإجابة بلا !

الجافا سكربت أصلا مصمم من قبل شركه نتسكيب . اذا نتسكيب أكثر دعما له . من جهة أخرى هناك فروقات أساسية في تعامل هذين المتصفحين مع الجافا ، وبعضها يطال حتى الأوامر الأساسيه.

سنتطرق لاحقا الى كيفية تجاوز هذه العقبة ، أما الآن فدعني أخبرك بأن أفضل طريقة للتأكد من عمل السكربت هو تجربته على كلا المتصفحين . طبعا ستدهشك كثرة المواقع التي فشلت في الوصول بصفحاتها الى أفضل شكل ودعم لكلا المتصفحين ، وهذا مايفسر السطر الذي نراه كثيرا:

لأفضل عرض ، ينصح بإستخدام المتصفح الفلاني.

طبعا ولأن الأغلبية تستخدم الإكسبلورر ، فسيكون التركيز علية في الشرح والأمثله ، ولن أنسى التطرق الى كيفية معرفة نوع المتصفح الخاص بالمستخدم.

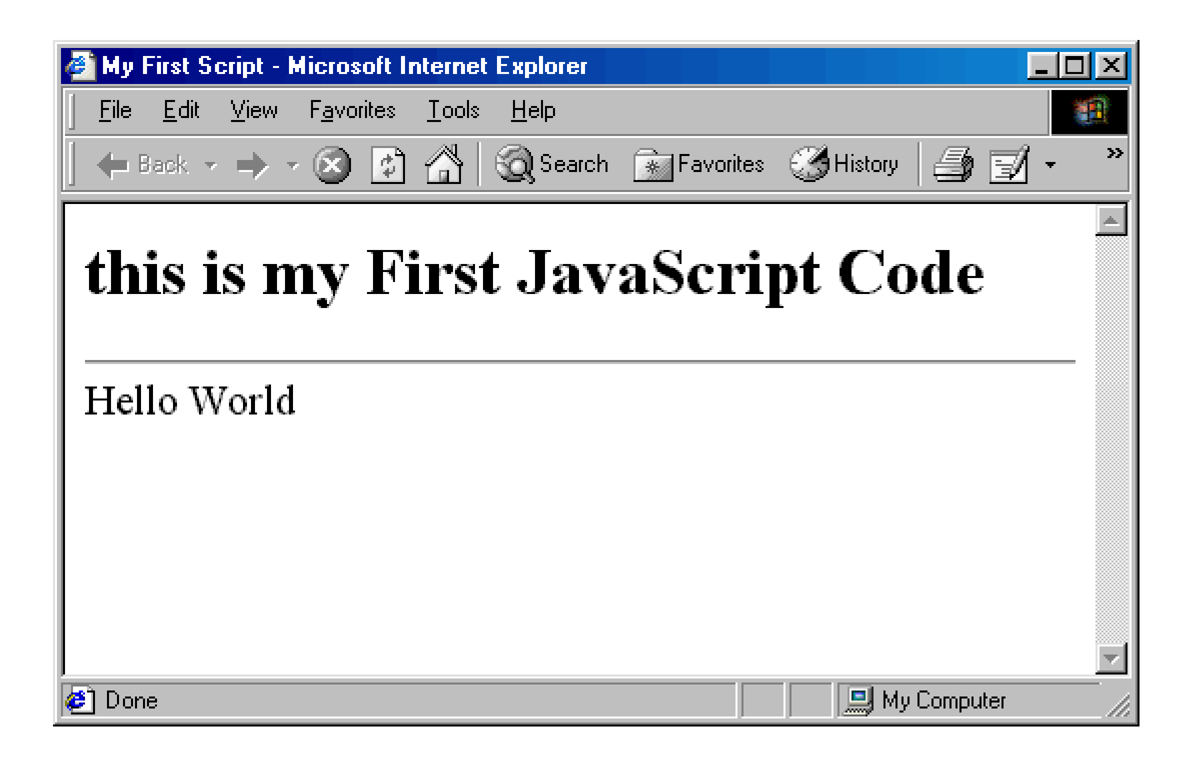
برنامجك الأول "Hello World"

هذا المثال الشهير بـ Hello World يمكننا من إستعراض أساسيات كتابة كود جافا سكربت

```
<HTML>
  <HEAD>
    <TITLE>My First Script</TITLE>
  </HEAD>

  <BODY>
    <H2> this is my First JavaScript Code </H2>
    <HR>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
           document.write("Hello World");
    //-->
    </SCRIPT>
  </BODY>
</HTML>
```

ويكون هذا هو الناتج عند عرضه من خلال المتصفح أنترنت أكسبلورار



هذا المثال البسيط سوف يكون البوابة الأولي لدخول عالم البرمجة بلغة الجافا سكربت
ومنه سوف نتعرف علي الخطوات المتبعة لكتابة كود جافا سكربت وهي كما يلي :

- نلاحظ أننا أدرجنا كود الجافا سكربت بداخل أكواد(أوسمه) هتمل HTML ، لذلك مبدئيا نلاحظ أن حتي يتم تنفيذ أكواد الجافا سكربت سوف نحتاج إلي إدراج كود الجافا سكربت في ملف ذو إمتداد htm أو html (أو ملفات ديناميكية مثل ASP أو PHP)
لذلك سوف نحفظ الكود السابق في ملف وليكون firstJs.htm
- ثانيا ربما يتبادر إلي الذهن كيف يتم الفصل بين كود الهتمل وكود الجافا سكربت لاحظ الكود التالي

```
<SCRIPT LANGUAGE="JavaScript">
        // هنا يتم كتابة كود الجافا سكربت
</SCRIPT>
```

بكل سهولة توفر لنا لغة الهتمل أحد الأوسمة (Tag) وهو <script>
هذا الوسم يحتاج ان نحدد له اسم اللغة المستخدمة لكتابة السكربت
ويتم هذا من خلال إستخدام المعامل Language بان نحدد له القيمة JavaScript
مع ملاحظة أنها هي القيمة الأفتراضية لذلك عادة لا نحتاج إلي تحديد لغة الأسكربت إلي جافا سكربت كما يلي

```
<SCRIPT>
        // كما تري لا نحتاج لتحديد لغة الأسكربت
        // هنا يتم كتابة كود الجافا سكربت
</SCRIPT>
```

ويتم وضع الكود الخاص بلغة الجافا سكربت بين الوسمين <Script> و </Script> كما يلي

```
<SCRIPT LANGUAGE="JavaScript">
<!--
        document.write("Hello World");
//-->
</SCRIPT>
```

- أذا كل ما نحتاجه الأن معرفة كيف يتم كتابة كود الجافا سكربت نعم هذا صحيح ولكن تمهل قليل وفكر معي ماذا يحدث لو كان المتصفح المستخدم لفتح الملف لا يدعم لغة الجافا سكربت ؟ أعتقد أنك سوف تقول بكل بساطة أن الكود المكتوب بهذه اللغة لن يعمل وهذا القول صحيح لكن سوف ينتظرك ما هو أسوء من هذا ألا وهو ظهور كود الجافا سكربت بداخل محتويات الصفحة كما بالشكل التالي



لكن لا تحزن يمكنك إخفاء ظهور كود الجافا سكربت بوضع الكود بين <!-- //--> كما يلي

```
<!--
        document.write("Hello World");
//-->
```

- الأن حان الوقت لمعرفة أول أمر في لغة الجافا سكربت وهو

```
document.write("Hello World");
```

document.write: يمكننا هذا الأمر من كتابة نص في المتصفح وبالتالي سوف يتم كتابة النص Hello World مع ملاحظة وضع النص المراد كتابتة بين علامتين تنصيص كما يلي

```
document.write("Hello World");
```

يجب ملاحظة التالي:
ان لغة الجافا سكربت لغة حساسة لحالة الحروف(الحروف الصغيرة والكبيرة) فعلي سبيل المثال document.write لا تكافيء document.Write ولو استخدمت بهذا الشكل سوف تؤدي لحدوث خطأ ولن يتم عرض النص Hello World

الفصل الثاني

أساسيات لغة الجافا سكربت

سوف نناقش إن شاء الله في هذا الفصل النقاط التالية :

- إدراج كود الجافا سكربت في صفحة هتمل
  - إدراج داخلي
  - إدراج خارجي

- التعليقات **Comments**
  - تعليق لسطر واحد
  - تعليق لسطر أو لأكثر من سطر

- طرق إدخال و إخراج البيانات
  - طرق إخراج البيانات
    - الطريقة alert
    - الطريقة write
    - الطريقة writeln
    - الطريقة print
  - طرق إدخال البيانات
    - الطريقة confirm
    - الطريقة prompt

إدراج كود الجافا سكربت في صفحة هتمل

كما رأينا سابقا في مثال Hello World أن كود الجافا سكربت غالبا ما يتم إدراجه بين كود الهتمل ويتم هذا الإدراج عن طريق :

١. **إدراج داخلي**
٢. **إدراج خارجي**

إدراج داخلي :

فيه يتم كتابة كود الجافا سكربت:

أ- إما بين الوسم <Script> ويمكنك وضع هذا الوسم بين الوسم <Head> أو الوسم <Body> أو كلاهما كما يلي

وضع كود الجافا سكربت بين وسم <Head>

```
<HTML>
    <Head>
       <script>
           <!--
              document.write("hello world");
           //-->
       </script>
    </Head>

    <Body></Body>
</HTML>
```

وضع كود الجافا سكربت بين وسم <Body>

```
<HTML>
    <Head></Head>

    <Body>
       <script>
           <!--
              document.write("hello world");
           //-->
       </script>
    </Body>
</HTML>
```

لكن ربما يتبادر لك السؤال التالي :

هل هناك فرق بين إدراج كود الجافا سكربت بين الوسم <Head> والوسم <Body> ؟

الإجابة : بالطبع نعم ولكن هذا الفرق ينحصر فقط في تسلسل تنفيذ الكود أي أن كود الجافا سكربت المندرج بين الوسمين <Head> سوف يتم تنفيذه أولا (حتي قبل تحميل عناصر الصفحة)

ثم يتم تنفيذ كود الجافا سكربت المندرج بين الوسمين <Body>

لاحظ معي المثال التالي

```
<HTML>
  <HEAD>
    <TITLE> الفرق بين وضع الكود </TITLE>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
           document.write(" مرحبا بك في الوسم Head");
           document.write("<br>");
    //-->
    </SCRIPT>
  </HEAD>

  <BODY>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
           document.write("  مرحبا بك في وسم Body");
    //-->
    </SCRIPT>
  </BODY>
</HTML>
```

كما تري تم وضع الكود التالي بين الوسم <Head>
وبالتالي سوف يتم تنفيذه أولا

```
    <SCRIPT LANGUAGE="JavaScript">
    <!--
           document.write(" مرحبا بك في الوسم Head");
           document.write("<br>");
    //-->
    </SCRIPT>
```

ثم وضع الكود التالي بين الوسم <Body>
وبالتالي سوف يتم تنفيذه بعد الأنتهاء من تنفيذ الكود السابق

```
    <SCRIPT LANGUAGE="JavaScript">
    <!--
           document.write("  مرحبا بك في وسم Body");
    //-->
    </SCRIPT>
```

ويكون الناتج كما يلي



ب- أو يتم وضع كود الجافا سكربت بداخل اي وسم هتمل كما يلي

```
<TagName event="يتم هنا إدراج كود الجافا سكربت"></TagName>
```

يتم إستبدال TagName بـاسم الوسم المراد وليكن علي سبيل المثال Span
ويتم إستبدال event بـاسم الحدث المطلوب وليكن علي سبيل المثال onclick
ثم يتم وضع الكود داخل هذا الحدث

وهذا النوع من إدراج كود الجافا سكربت سوف نتناوله في الجزء الثاني من الكتاب بشكل مفصل ولكن دعنا نري مثال سريع يوضح الطريقة كما يلي

```
<HTML>
  <HEAD>
    <TITLE> إدراج الكود داخل الأوسمة </TITLE>
  </HEAD>

  <Body>
     <Span onclick="document.write('نحن الأن بالداخل')"> قم بالضغط علي هذا النص </Span>
  </Body>
</HTML>
```

تم إستخدام الحدث onclick (اي عند الضغط علي محتويات الوسم المختار وهو هنا Span) كما يلي

```
<Span onclick="document.write('نحن الأن بالداخل')"> قم بالضغط علي هذا النص </Span>
```

فعند الضغط علي محتويات الوسم <Span> سوف يتم تنقيذ الكود التالي

document.write('نحن الأن بالداخل')

```
<Span onclick="document.write('نحن الأن بالداخل')"> قم بالضغط علي هذا النص </Span>
```

إدراج خارجي :

فيه يتم إدراج كود الجافا سكربت من ملف خارجي غالبا ما يكون ذو إمتداد js
كما يلي

```
<SCRIPT LANGUAGE="JavaScript" src="fileName.js"></SCRIPT>
```

ويتم كتابة كود الجافا سكربت في الملف ذو الإمتداد js

مثال توضيحي : يتكون من ملفين أحدهما
ExternalEmbed.htm يدرج به الملف outerScript.js الذي يحتوي علي كود الجافا سكربت

ExternalEmbed.htm

```
<HTML>
    <Head>
        <SCRIPT LANGUAGE="JavaScript" src="outerScript.js"></SCRIPT>
    </Head>

    <Body>
           إدراج خارجي
    </Body>
</HTML>
```

outerScript.js

```
document.write("هذا نص خارجي من الملف المدرج");
document.write("<br>");
```

التعليقات (Comments)

تخيل معي أنك قمت بكتابة كود جافا سكربت معقد وطويل (مثلا حوالي ٢٠٠٠ سطر) في مثل هذه الحالات سوف تواجه التالي :

- صعوبة قراءة الكود
- صعوبة تصحيح الأخطاء Debug

إذا ما الحل :
من الخطوات المتبعة في تقليص هذة المشكلة عمل ملاحظات أو تعليقات Comments لكل جزء من الكود
وكما سنري يوجد نوعين من التعليقات :

- **تعليق لسطر واحد (//)**

بإستخدام العلامتين التاليتين // ثم يليهم تعليقك الخاص بك كما بالمثال التالي :

```
<HTML>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
           // إظهار رسالة ترحيب للمستخدم
           document.write("مرحبا بكم");
           document.write("<p>");   // طباعة سطر فارغ
    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

- **تعليق لسطر أو أكثر من سطر (/* */)**

بإستخدام العلامتين التاليتين */ ثم يليهم تعليقك الخاص بك ربما يكون مكون من سطر أو أكثر ثم نضع مرة اخرة العلامتين /* لإنهاء التعليق
كما بالمثال التالي :

```
<HTML>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
           /*
              إظهار رسالة ترحيب للمستخدم
              ثم
              طباعة سطر فارغ
           */
           document.write("مرحبا بكم");
           document.write("<p>");   /* طباعة سطر فارغ */
    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

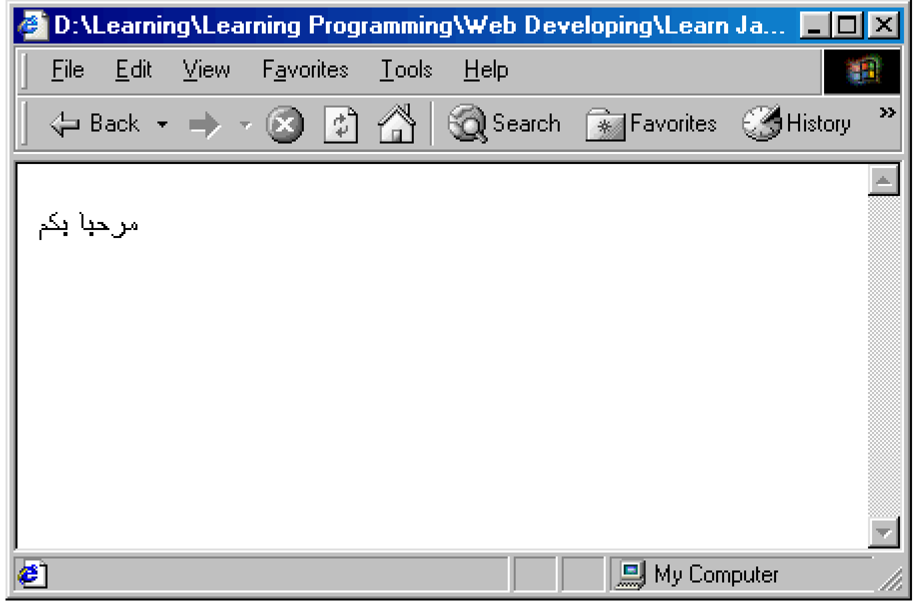
يمكنك إستخدام اي نوع من التعليقات أو كلاهما كما يفضل المبرمج.
كلا النوعين من التعليقات يتم تجاهله من قبل مفسر لغة الجافا سكربت (JavaScript Interpreter) إي لا يتم تنفيذه علي سبيل المثال ربما تقوم بالتالي
تحويل كود مكون من سطر أو أكثر إلي تعليق (تستخدم هذة الطريقة في عملية تصحيح الأخطاء)
وبالتالي لن يتم تنفيذه كما بالمثال التالي :

```
<HTML>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
           /*
              إظهار رسالة ترحيب للمستخدم
              ثم
              طباعة سطر فارغ
           */
           document.write("مرحبا بكم");
           document.write("<p>");
           // document.write("إلي اللقاء");
    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

كما تري لن يتم تنفيذ السطر التالي

```
           // document.write("إلي اللقاء");
```

ويكون الناتج كما بالشكل التالي



ملاحظة هامة :

لا يمكنك عمل تداخل (nesting) عند إستخدامك للنوع الثاني من التعليقات تعليق لسطر أو أكثر من سطر (/* */)
لإن ذلك سوف يؤدي إلي إحداث خطاء لغوي Syntax Error
لاحظ معي الأشكال التالية من الأخطاء :

```
<HTML>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
           /*
              إظهار رسالة ترحيب للمستخدم
              ثم
             /* طباعة سطر فارغ */
           */
           document.write("مرحبا بكم");
           document.write("<p>");
    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

```
           /*
              إظهار رسالة ترحيب للمستخدم
              ثم
             /* طباعة سطر فارغ */
           */
```

```
           /*
              إظهار رسالة ترحيب للمستخدم
              ثم
             /* طباعة سطر فارغ
                    /* تداخل جديد */
              */
           */
```

طرق إدخال و إخراج البيانات

توفر لنا اي لغة برمجة طرق لإدخال وإخراج البيانات وهي من النقاط الهامة التي تعطي برنامجك تفاعل مع المستخدمين لبرنامجك
وسوف نتناول في هذا الجزء الطرق المختلفة لإخراج وإدخال البيانات .

أولا : طرق إخراج البيانات

عندما يقال أن لغة الجافا سكربت توفر لنا طرق لإخراج البيانات نعني بذلك أن لغة الجافا سكربت توفر لك بعض الأوامر الخاصة بها التي تمكنك من إظهار رسائل تظهر من خلال وحدات الإخراج بجهاز الكمبيوتر (مثل الشاشة والطابعة) وسوف نتناول هذة الطرق المختلفة .

١- الطريقة alert :

- تقوم بإظهار رسالة إلي المستخدم .
- تعتبر إحدي الوظائف التابعة للكائن window، كما سنري لاحقا
- طريقة إستخدامها :

```
alert(" ضع هنا رسالتك ");
```

أو

```
window.alert(" ضع هنا رسالتك ");
```

يتم كتابة الأمر alert أو window.alert كلاهما سوف يؤدي إلي نفس النتيجة ثم يتم تمرير الرسالة المراد إظهارها بين علامتين تنصيص ""

تمرين لإظهار رسالة تحذير للمستخدم

```
<HTML>
  <Title> الطريقة alert </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
           alert(" من فضلك يجب إدخال بيانات صحيحة");
    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

ويكون الناتج كما يلي :



٢- الطريقة write :

- تقوم بكتابة نص في الصفحة للمستخدم .
- تعتبر إحدي الوظائف التابعة للكائن document، كما سنري لاحقا
- طريقة إستخدامها :

```
document.write(" ضع هنا رسالتك ");
```

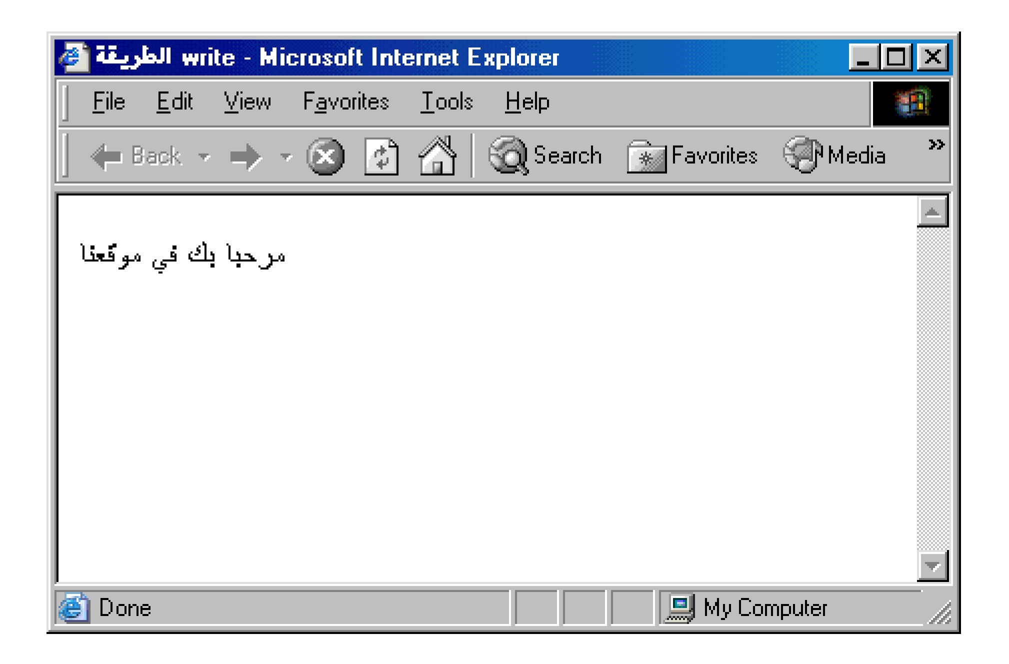
أيضا يتم تمرير الرسالة المراد كتاباتها بين علامتين تنصيص "" وتكون هذة الرسالة إما نص بسيط plain text أو نص هتمل HTML text

يجب ملاحظة أن لغة الجافا سكربت لغة حساسة لحالة الحروف لذلك يكتب الأمر كما يلي write جميع حروفه صغيرة (small letters)

تمرين لإظهار رسالة ترحيب بسيطة للمستخدم

```
<HTML>
  <Title> الطريقة write </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
           document.write("مرحبا بك في موقعنا");
    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

ويكون الناتج كما يلي :



تمرين لإظهار رسالة تأكيد لعملية حذف للمستخدم

في هذا التمرين سوف نتعلم طريقة إستخدام الأمر document.write

- لإظهار رسالة أكثر تعقيد منصقة بإستخدام جمل الهتمل .
- طريقة دمج النصوص .
- طريقة حل المشكلة الناتجة من تداخل علامات التنصيص .

```
<HTML>
  <Title> الطريقة write </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
           document.write("<Font Color=red>هل أنت متأكد من إتمام عملية الحذف</Font>");
           document.write ("<P>");
           document.write("<Center>" + "<Input type=button value=موافق> ");
           document.write("<Input type=button value='غير موافق'></Center>");
           document.write("</P>");
    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

ويكون الناتج كما يلي :



كما ذكرنا في النقاط المستفادة من هذا التمرين سنلاحظ التالي :

- طريقة دمج النصوص:

يتم دمج النصوص (والنص هو الجملة المحددة بعلمتين تنصيص) بإستخدام الرمز + المستخدم في عمليات الجمع الرياضي كما يلي

```
           document.write("<Center>" + "<Input type=button value=موافق> ");
```

ـ علاج تداخل علامات التنصيص :

كما ذكرنا سابقا أن النص هو عبارات وجمل يتم كتابتها بين علامتين تنصيص ""
علي سبيل المثال نريد عمل نص به عبارة أنا من محبي لغة الجافا سكربت سوف نقوم بوضعها بين علامتين التنصيص كما يلي " **أنا من محبي لغة الجافا سكربت** " وبذلك يستطيع مفسر اللغة تميز أن هذه الكلمات تابعة لنص واحد .
إذا إين المشكلة ومتي تظهر نعم أنت علي حق لا توجد مشكلة هنا إلا إذا حدث التالي
تخيل معي أن نص الجملة يحتوي علامة تنصيص أو أكثر
علي سبيل المثال أنك تريد كتابة الجملة التالية (**هل هناك** "**مشكلة**" **يا رجل ؟**) كما تري أننا الأن في مأزق لأننا لو وضعنا هذة الجملة بين علامتيين تنصيص سوف يحدث تداخل في علامات التنصيص وسوف يؤدي هذا إلي إرتباك لمفسر اللغة مما ينتج عنه خطأ لغوي syntax error .
الأن ما الحل ؟
ربما يتبادر إلي ذهنك الهرب من المشكلة وتقول أنا لست في حاجة لإظهار علامات التنصيص في الجملة وسوف أجعل الجملة بدونهما كما هو حالنا نحن العرب ولكن دائما تأتي الحلول لتفادي الأخطاء وتجنبها وليس الهرب منها ، أعتقد أنه تبادر إليك الأن أن هذه المشكلة لها حل ابشرك بقولي نعم
حيث توفر لنا لغة الجافا سكربت علامات الهروب Escaping Characters .

علامات الهروب Escaping Characters :

تمكنك من تضمين بعض الحروف التي يصعب كتابتها في محتوي النصوص ومنها التالي :

\'	تمكنك من إضافة علامه التنصيص الفردية بداخل النص
\"	تمكنك من إضافة علامه التنصيص بداخل النص
\r	تمكنك من إدخال حرف خاص لعمل ما يسمي Carriage return اي تراجع موشر الكتابه إلي بداية السطر مما ينتج عنه في بعض الأحيان عمل سطر جديد
\n	تمكنك من عمل سطر جديد داخل النص
\t	تمكنك من إدخال الحرف Tab

بعد التعرف السريع لعلامات الهروب كيف يمكننا حل مشكلة كتابة النص السابق (هل هناك "مشكلة" يا رجل ؟)
بكل يسر يتم إستبدال اي علامة تنصيص (") بي علامة الهروب (\") ويكون النص كالتالي
(هل هناك \"مشكلة\" يا رجل ؟)

مثال تطبيقي لكتابة الرسالة التالية :

نتمنا لك يا قارئنا "العزيز" أن تكون قد
تعلمت ما سبق من قول و ان تحرص علي ما هو أت

```
<HTML>
  <Title> الطريقة write و alert </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
           alert("نتمنا لك يا قارئنا \"العزيز\" أن تكون قد " + "\r\n" +
                     "تعلمت ما سبق من قول و ان تحرص علي ما هو أت ");

           document.write("نتمنا لك يا قارئنا \"العزيز\" أن تكون قد " + "<br>" +
                     "تعلمت ما سبق من قول و ان تحرص علي ما هو أت ");
    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

ويكون الناتج كالتالي





ملاحظة :

لكتابة سطر جديد

بالطريقة alert : قمنا بإستخدام علامات الهروب **\r\n** لعمل سطر جديد

بالطريقة write: قمنا بإستخدام الوسم **
** حتي نتمكن من كتابة سطر جديد

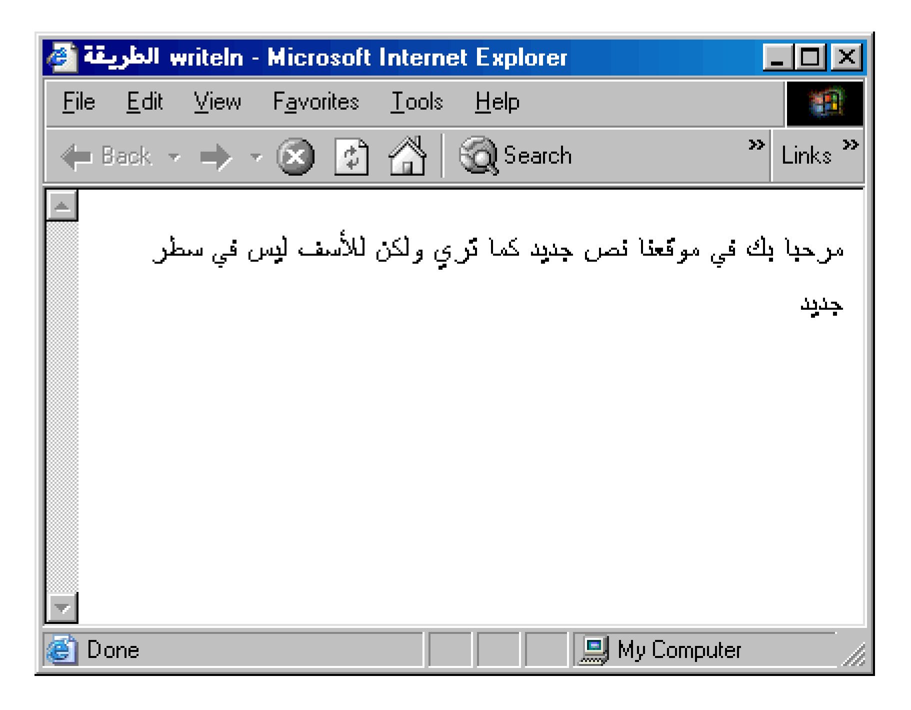
writeln

- تقوم بكتابة نص في الصفحة للمستخدم بالإضافة إلي أنها تحدث سطر جديد قبل كتابة النص.
- تعتبر إحدي الوظائف التابعة للكائن document، كما سنري لاحقا
- طريقة إستخدامها :

```
document.writeln(" ضع هنا رسالتك ");
```

أيضا يتم تمرير الرسالة المراد كتابتها بين علامتين تنصيص "" وتكون هذة الرسالة إما نص بسيط plain text أو نص هتمل HTML text

```
<HTML dir=rtl>
  <Title> الطريقة writeln </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
           document.write("مرحبا بك في موقعنا ");
           document.writeln("نص جديد كما تري ولكن للأسف ليس في سطر جديد");
    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```



كما تري لم يتم عمل سطر جديد أقول لك نعم أننا لا نري سطر جديد ولكن هذا لا يعني عدم وجودة حيث أنه تم عمل تغذيه لسطر جديد باستخدام \n وكما نعلم أن لغة الهتمل تتجاهل المسافات الزائدة والسطور الجديدة .

٣- الطريقة print :

- تقوم بطباعة الصفحة .
- تعتبر إحدي الوظائف التابعة للكائن **window** .
- طريقة إستخدامها :

```
window.print();
```

أيضا يوجد حدثان onBeforeprint و onAfterprint سوف نتحدث عنهما فيما بعد.

ثانيا : طرق إدخال البيانات

١- الطريقة confirm :

- تقوم بإظهار رسالة مثل الطريقة **alert** السابقة بالإضافة إلي أنها تقوم بإرجاع أحدي القيمتين true أو false.
- تعتبر إحدي الوظائف التابعة للكائن **window** .
- طريقة إستخدامها :

```
confirm(" ضع هنا رسالتك ");
```

أو

```
window.confirm(" ضع هنا رسالتك ");
```

يتم كتابة الأمر confirm أو window.confirm كلاهما سوف يؤدي إلي نفس النتيجة ثم يتم تمرير الرسالة المراد إظهارها بين علامتين تنصيص ""
ثم يتم إستقبال القيمة الراجعة من الإختيار في متغير (سوف يتم شرح المتغيرات فيما بعد)

تمرين لإظهار رسالة تأكيد حذف إشتراك احدي العملاء

```
<HTML>
  <Title> الطريقة confirm </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
           confirm(" من فضلك هل أنت متأكد من إتمام عملية حذف هذا العميل ؟");
    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

ويكون الناتج كما يلي :



كما تري ظهور الرسالة ولكن بعد الضغط علي إحدي الزرارين cancel أو ok لم نتعرف أيهما تم الضغط عليه ، وحتي نتعرف علي القيمة الراجعة من الإختيار نحتاج إلي :

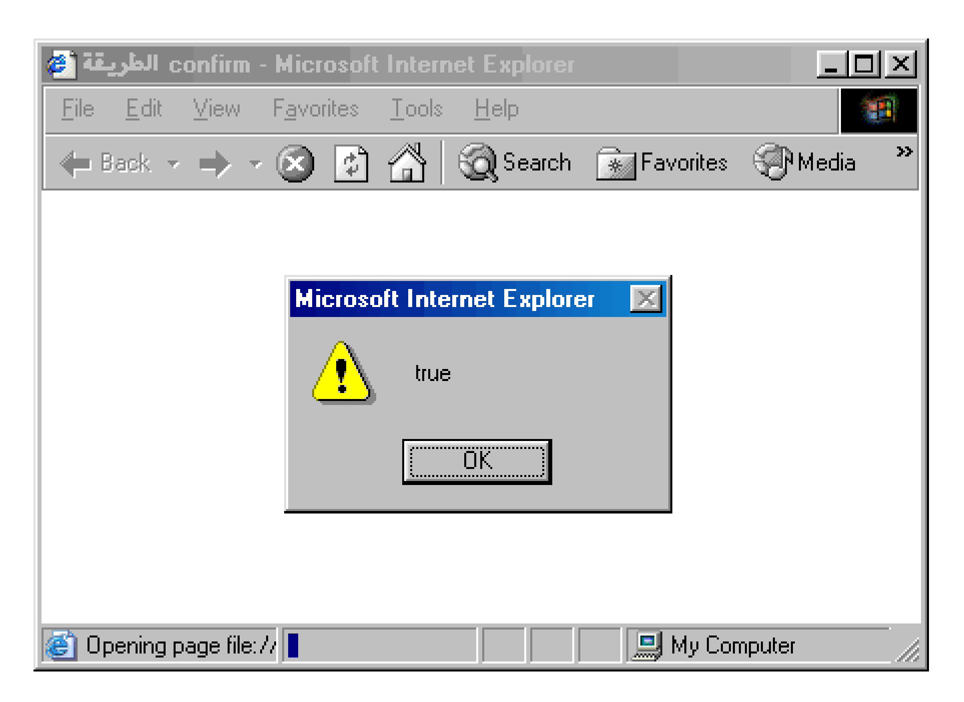
- تخزين القيمة الراجعة في متغير ثم نحدد قيمة هذا المتغير فإذا كانت قيمته true فهذا يعني أن المستخدم قد ضغط الزر ok ، وإذا كانت قيمته false فهذا يعني أن المستخدم قد ضغط زر cancel وسوف نتحدث علي هذا الإسلوب فيما بعد عند الحديث عن المتغيرات.
- أو نقوم بإستخدام القيمة الراجعة بدون تخزينها (من علي الطائر كما يقال) كما سوف يتضح في المثال التالي : حيث سوف نقوم بطباعة القيمة الراجعة من الإختيار بواسطة الطريقة alert كما يلي :

```
<HTML>
  <Title> الطريقة confirm </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
           alert( confirm(" من فضلك هل أنت متأكد من إتمام عملية حذف هذا العميل ؟") );
    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

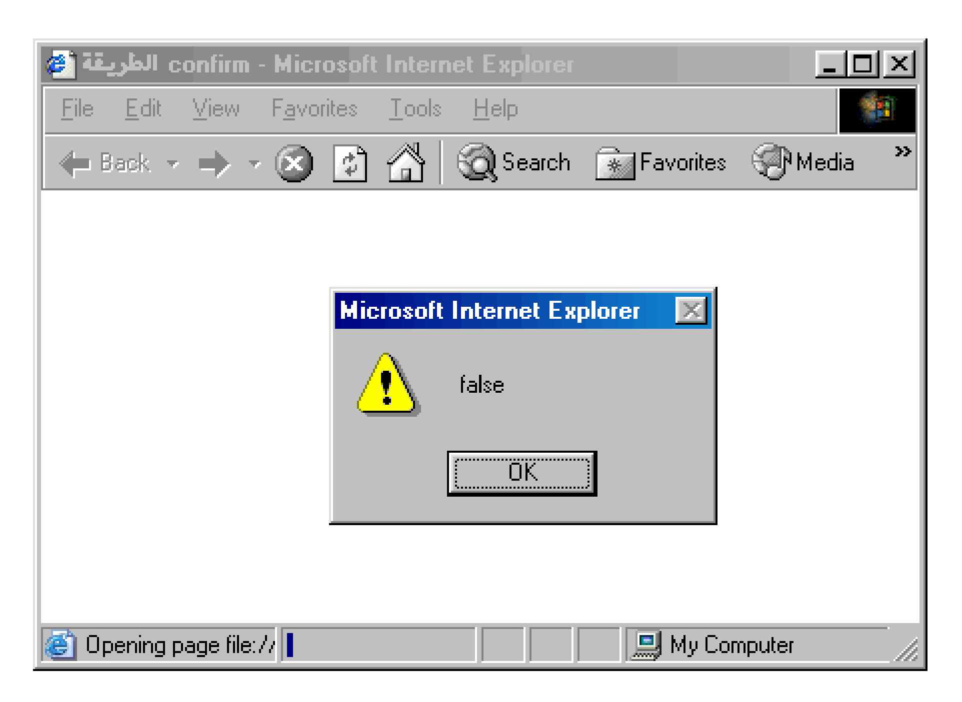
ويكون الناتج كما يلي :



فعند الضغط علي زر ok يكون الناتج كالتالي :



وعند الضغط علي زر cancel يكون الناتج كالتالي :



٢- الطريقة prompt :

- تقوم بإظهار رسالة مثل الطريقة **confirm** السابقة بالإضافة إلي أنها تقوم بإرجاع أحدي القيمتين القيمة المدخلة من خلال المستخدم أو القيمة null (وهي تعني لا شيء).
- تعتبر إحدي الوظائف التابعة للكائن **window** .
- طريقة إستخدامها :

```
prompt(" ضع هنا رسالتك ", "القيمة الإفتراضية للقيمة الراجعة");
```

أو

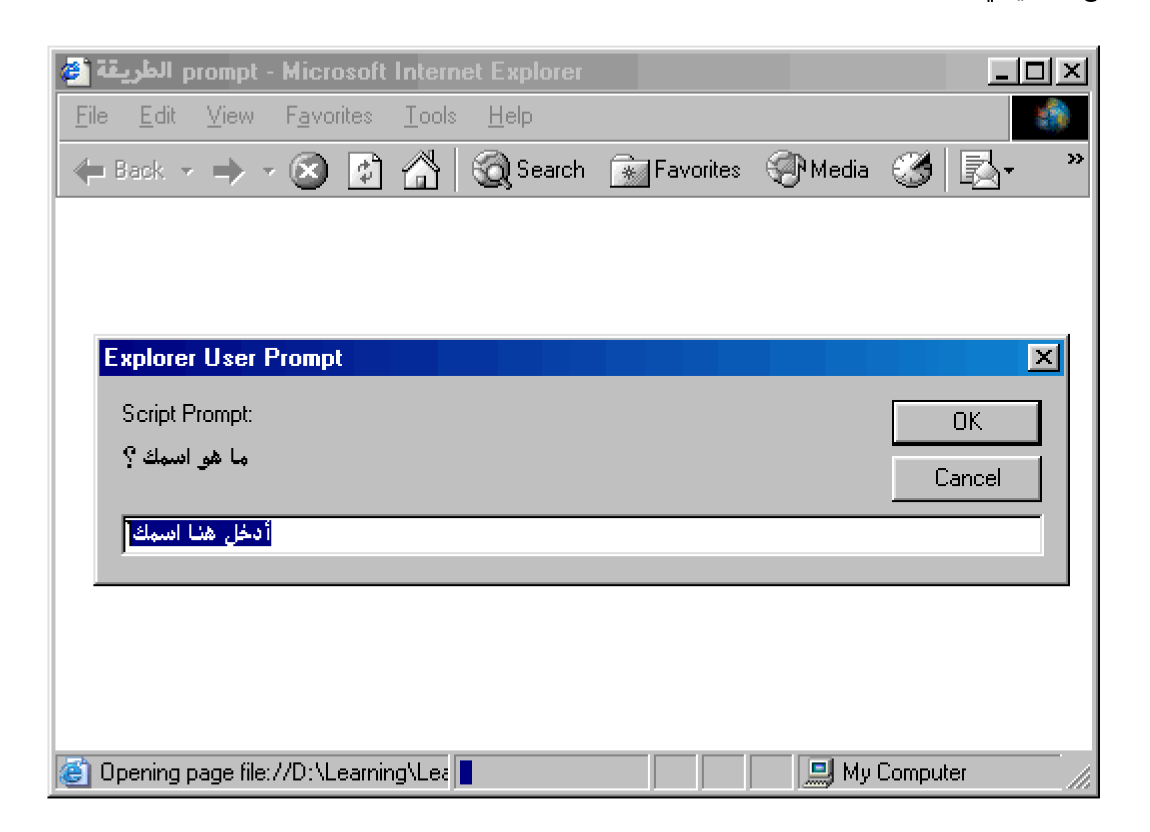
```
window.prompt(" ضع هنا رسالتك ", "القيمة الإفتراضية للقيمة الراجعة");
```

يتم كتابة الأمر prompt أو window.prompt كلاهما سوف يؤدي إلي نفس النتيجة ثم يتم تمرير الرسالة المراد إظهارها بين علامتين تنصيص "" ثم تمرر أو لا تمرر القيمة الإفتراضية للقيمة الراجعة وإذا لم تمرر سوف تأخذ القيمة الإفتراضية undefined
ثم يتم إستقبال القيمة الراجعة من الإختيار في متغير (سوف يتم شرح المتغيرات فيما بعد) أو إظهارها في شكل رسالة كما سبق القول في الطريقة confirm كما بالمثال التالي

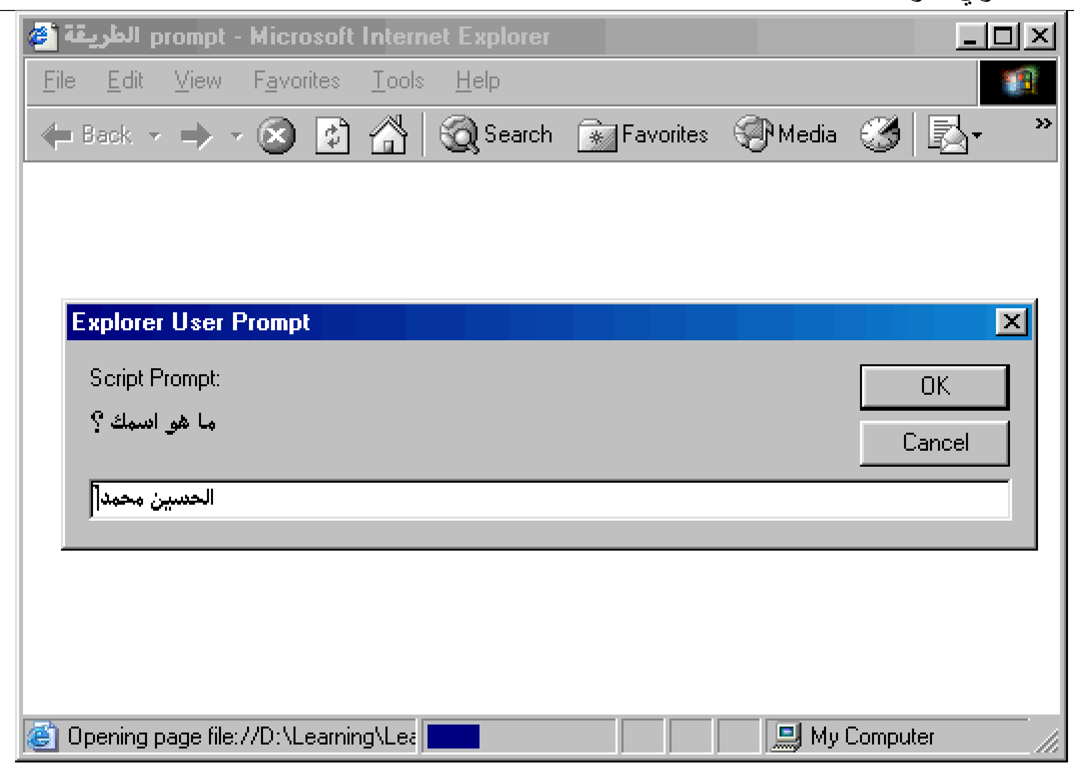
تمرين لإظهار رسالة بها اسم المستخدم بعد إدخال اسمه

```
<HTML>
  <Title> الطريقة prompt </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
           alert( prompt(" ما هو اسمك ؟", "أدخل هنا اسمك") );
    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

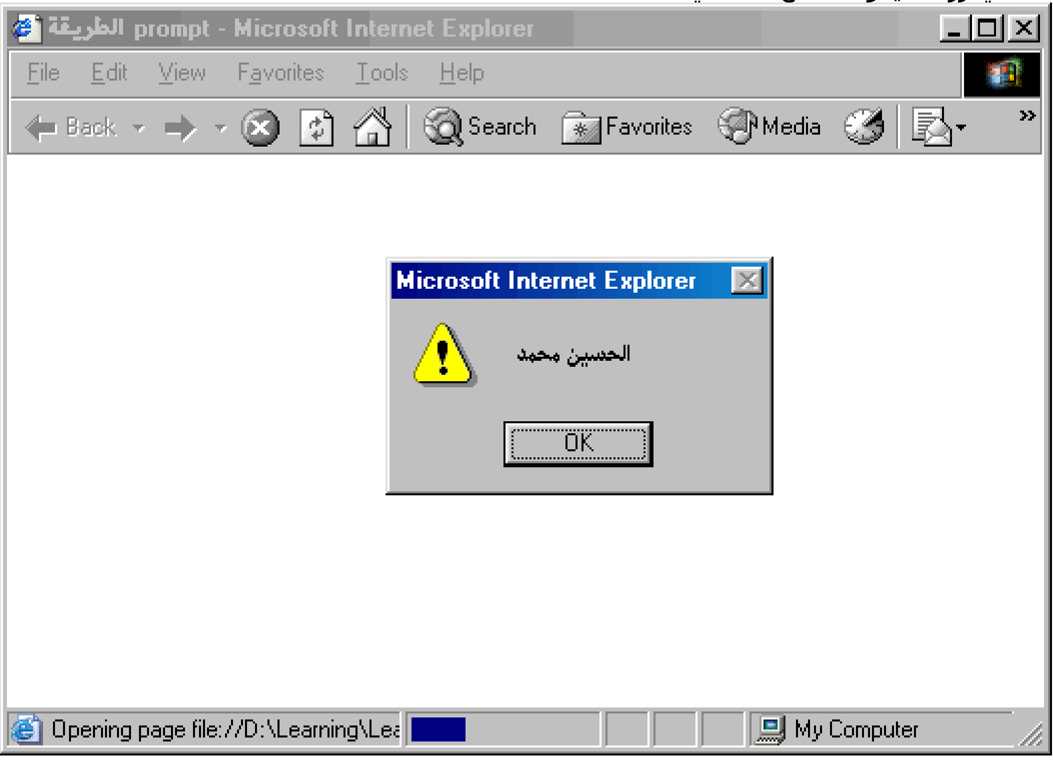
ويكون الناتج كما يلي :



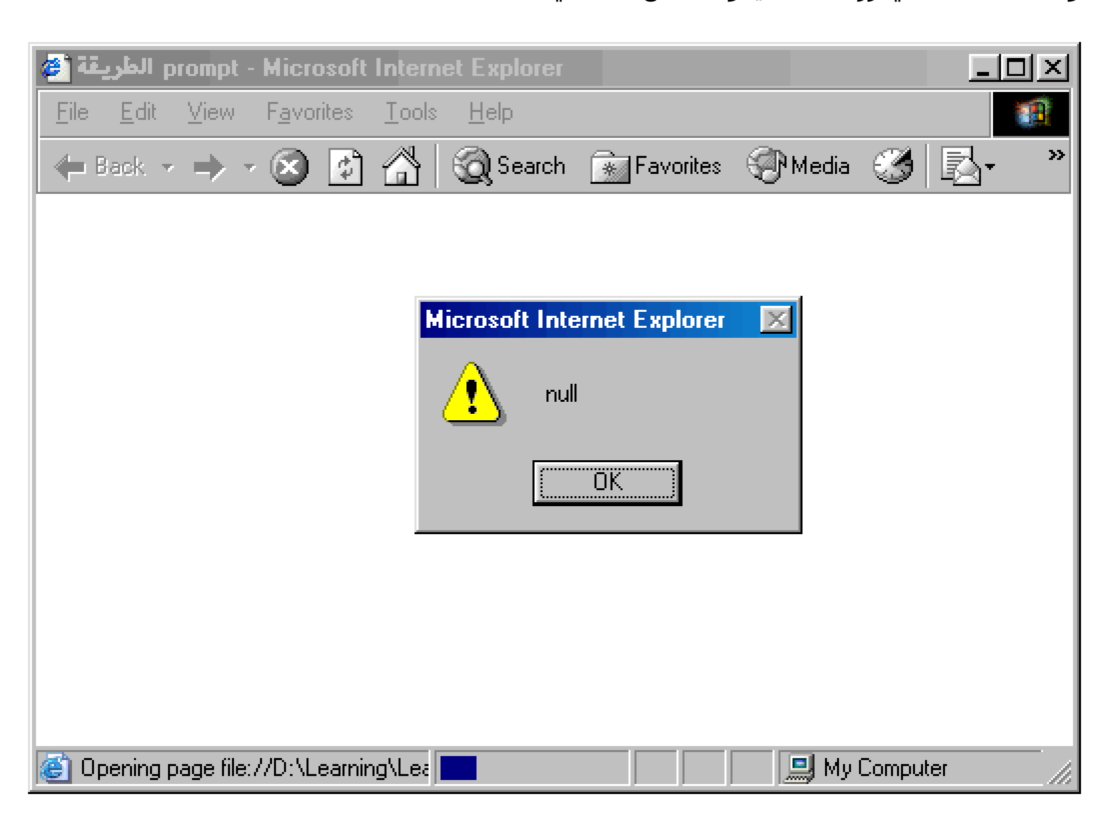
ثم نقوم بإدخال الاسم وليكن التالي "الحسين محمد" كالتالي :



فعند الضغط علي زر ok يكون الناتج كالتالي :



وعند الضغط علي زر cancel يكون الناتج كالتالي :



الفصل الثالث

المتغيرات

سوف نناقش إن شاء الله في هذا الفصل النقاط التالية :

- المتغيرات **Variables**
  - ما هو المتغير
  - أنواع البيانات
  - طريقة تسمية المتغيرات
  - تعريف المتغيرات
- الكلمات المحجوزة **JavaScript Keywords**

المتغيرات Variables

ما هو المتغير :

هب أنني أعطيتك كمية من الماء فما وسيلة تخزينك لهذة الكمية ، ربما قلت لي سوف أضعها في كوب صغير أو إناء أو أي محتوي أكبر ، وبهذا أقول لك أنك نجحت في عملية تخزين المياه ولكن ما فائدة هذة العملية ، ربما تقول لي حتي أستخدمها عند حاجتي لها في المستقبل القريب أو البعيد نعم نعما الرأي ما قلت .

لكن ما علاقة هذا بموضوعنا المتغيرات ، بكل بساطة كما أحتجنا إلي تخزين الماء في محتوي يحفظه حتي نتمكن من إستخدامه عند الحاجة إليه ، فإننا أيضا في عالم الكمبيوتر والبرمجة نحتاج إلي عملية تخزين ولكن ليس للماء ولكن **تخزين للبيانات** .

فهب أنني أعطيتك النص التالي **الحمد لله العلي العظيم** وقلت لك كيف يمكنك تخزين هذا النص ربما أسرعت بالقول لي أنني سوف أقوم بحفظ هذا النص في رأسي وتكون رأسك هنا هي مخزن النص ، ولكن إذا أردنا تخزين النص السابق في برنامجنا بشكل مؤقت (اي خلال فترة عمل البرنامج فقط وينتهي تخزينها عند إنتهاء البرنامج) هنا سوف نحتاج إلي مخزن يتم حفظ هذا النص به وغالبا ما يكون جزء من ذاكرة الجهاز المؤقتة ، ويتم إعطاء هذا الجزء من ذاكرة الجهاز اسم خاص حتي يتم التعامل مع هذا الجزء .

نخلص مما سبق إلي أن **المتغير** هو جزء من ذاكرة الجهاز يتم إعطاء اسم له ، يعتبر بمثابة مخزن للبيانات

أنواع البيانات :

تختلف أنواع البيانات فمثلا يستطيع الإنسان إدراك التميز بين الحروف الأبجدية و الأرقام وبين النصوص والقيم العددية وهكذا ، وأيضا توفر لنا لغة الجافا سكربت التميز بين ٤ أنواع من البيانات :

- النصوص String
- القيم العددية Number
- الحالة البولينية (صح أم خطأ) Boolean
- القيمة لا شئ Null

لاحظ معي التالي كما نعلم أن شكل وطبيعة تركيب مخازن المياه يختلف عن بنية مخازن البترول مثلا وهكذا يختلف نوع المخزن حسب طبيعة ونوع المادة المخزنة به .

ولكن **المتغيرات** في لغة الجافا سكربت (وهي مخازن البيانات) لا تختلف في بنيتها بإختلاف نوع البيانات المخزنة بها كما هو الحال بلغة الجافا والسي التي يتميز فيها نوع المخزن حسب نوع البيانات المخزنة به .

طريقة تسمية المتغيرات :

تسمية المتغيرات تخضع لشروط أساسية يجب توافرها ليعمل البرنامج بالصورة الصحيحة وهي كالتالي :

- الخانة الأولي من إسم المتغير لابد أن تكون أحد الأحرف الإنجليزية سواءً حرفا كبيرا أو صغيرا مع ملاحظة الفرق بينهما ، ويمكن البدء بعلامة _ أو علامة $ ، مع أنه لاينصح بإستخدام العلامة الأخيرة.
- لايمكنك إطلاقا إستخدام رقم كأول خانة في الإسم.
- الأسماء لايمكن أن تحوي مسافات بين أحرفها ، لكن يمكنك إستخدام العلامة _ بدل المسافات.
- لايمكن لأي إسم أن يماثل أيا من الكلمات المحجوزة وهذه القاعدة عامة لجميع لغات البرمجة.

وإليك بعض الأمثلة على التسمية الصحيحة للمتغيرات :

- Address1
- A4xb5
- lastName
- _firstName
- parent_Name

وإليك بعض الأمثلة على التسمية غير الصحيحة للمتغيرات :

- 1stName
- ?subName
- last name
- @userID

جرت العادة عند تسمية المتغيرات في الجافا سكربت أنه لا يفضل إستخدام الحرف _ للفصل بين مقاطع الكلمات ولكن يتم دمج المقاطع مع جعل الحرف الأول صغير وأول حرف من المقاطع التالية يكون ذو حرف كبير فعلي سبيل المثال عندما نريد تسمية متغير ليعبر عن last name لا يفضل تسميته كما يلي : Last_name ولكن يفضل تسمية كما يلي lastName.

ملاحظات هامة عند تسمية المتغيرات :

كما نعلم فإن لغة الجافا سكربت تميز بين حالة الحروف الكبيرة والصغيرة فإن المتغير lastName ليس مثل المتغير LastName .

دائما حاول أن تكون أسماء المتغيرات ذات دلالة تبسط عليك في المستقبل عملية التعديل في البرنامج .

تعريف المتغيرات :

كما غلمنا أن المتغيرات هي أسماء لحجز أماكن في ذاكرة الجهاز لحفظ البيانات التي تسند إليها . وقبل أن تستخدم أيا من هذه المتغيرات لابد من الإفصاح عنها ، وذلك بإستخدام الأمر var

```
var firstName;
```

وهكذا ، كما أنه من الممكن الإعلان عن أكثر من متغير في سطر واحد ، وهذا مالايمكنك عمله بواسطة الفيجوال بيسك سكربت ..

```
var firstName, lastName, userID;
```

وأخيراً ، لمزيد من التبسيط ، يمكننا إسناد القيم الى هذه المتغيرات أثناء تعريفها والإفصاح عنها، كما يلي:

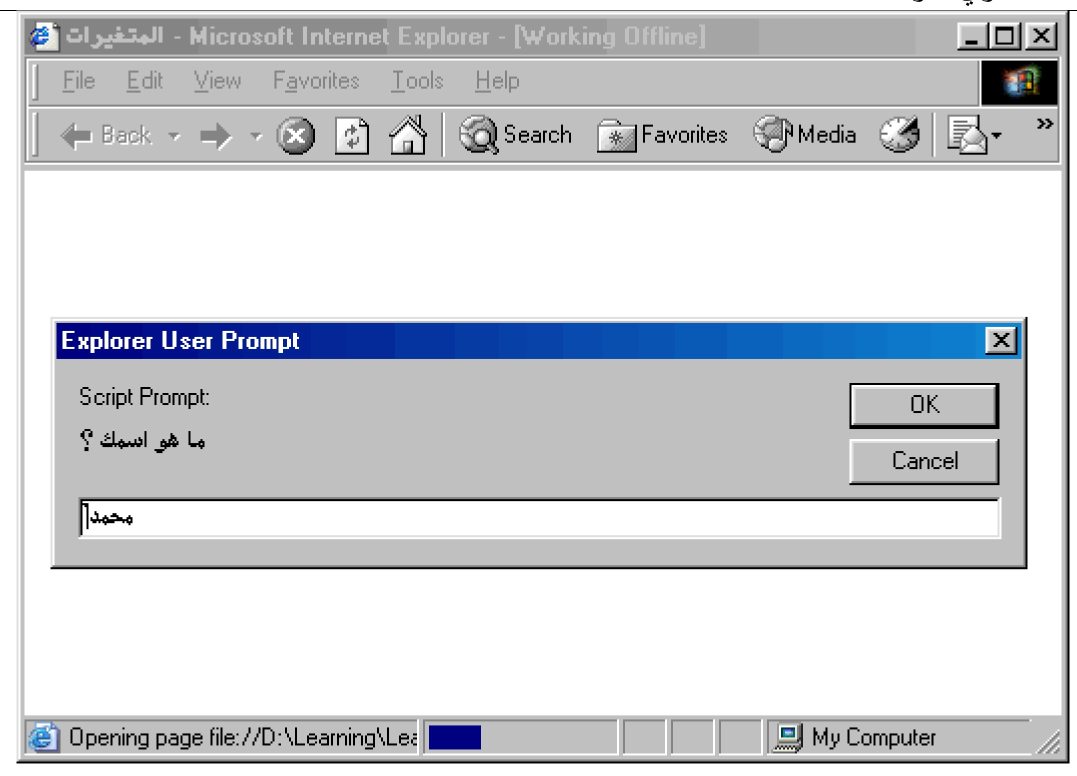
```
var firstName, lastName = "علي", userID = 13;
```

تذكر أنه إذا لم تفصح عن أي متغير قبل إستخدامه فإن ذلك سيوقف عمل البرنامج ، ولاتنسى أن الجافا سكربت لغة حساسة تجاه الأحرف كما ذكرنا سابقا فـ x غير X دائما حتى في أوامر الجافا السكربت والوظائف ومصطلحاتها المحجوزة.

تمرين لإظهار رسالة ترحيب للمستخدم بعد إدخال اسمه

```
<HTML>
  <Title> المتغيرات </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
           var username = prompt(" ما هو اسمك ؟", "أدخل هنا اسمك");
           alert( "مرحبا بك يا : " + username );
    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

ويكون الناتج بعد إدخال الأسم بقيمة محمد كما يلي :



وبعد الضغط علي الزر ok :

المتغيرات - Microsoft Internet Explorer - [Working Offline]
File Edit View Favorites Tools Help
Back Search Favorites Media
Microsoft Internet Explorer
مرحبا بك يا : محمد
OK
Opening page file://D:\Learning\Lea
My Computer

وكما نري فقد تم الإعلان عن متغير باسم userName ثم خزن به القيمة الراجعة من الأمر prompt كما يلي

```
var username = prompt(" ما هو اسمك ؟", "أدخل هنا اسمك");
```

ثم قمنا بطباعة القيمة المخزنة بالمتغير userName مضافا إليها نص " **مرحبا بك يا :** " كما يلي :

```
var username = prompt(" ما هو اسمك ؟", "أدخل هنا اسمك");
alert( "مرحبا بك يا : " + username );
```

تمرين لإظهار بيانات المستخدم بعد إدخال اسمه وعمره

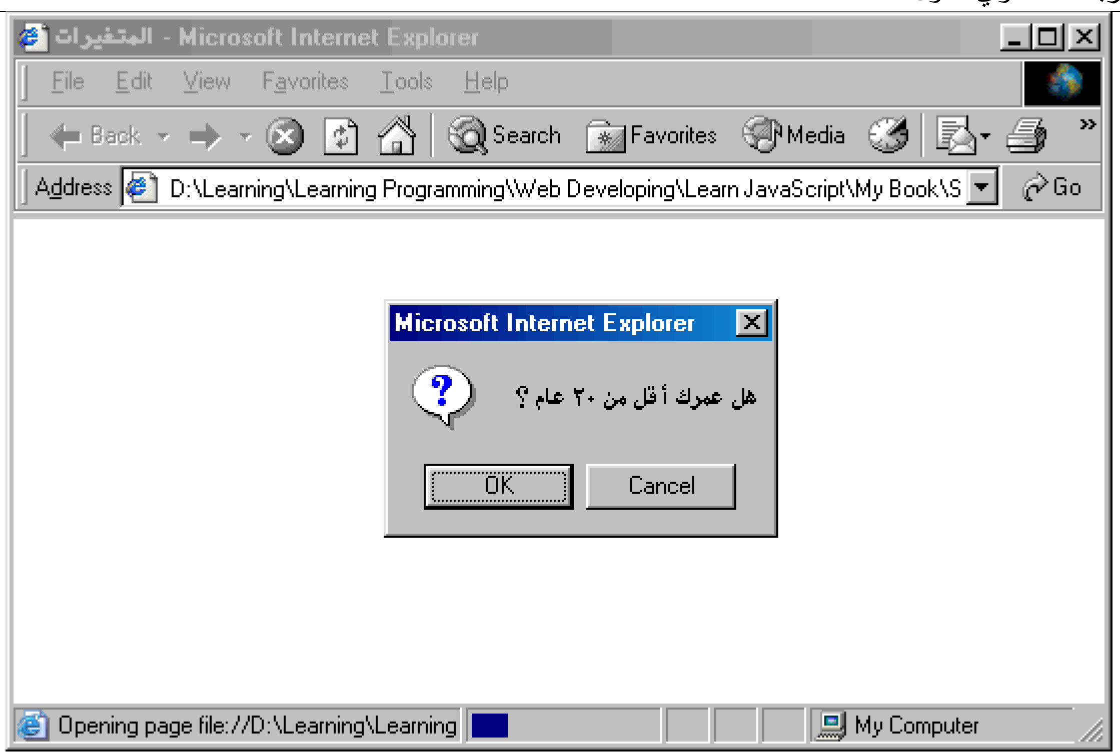
```
<HTML dir=rtl>
  <Title> المتغيرات </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
        var userage, username;

        userage = confirm("هل عمرك أقل من ٢٠ عام ؟");
        username = prompt(" ما هو اسمك ؟", "أدخل هنا اسمك");

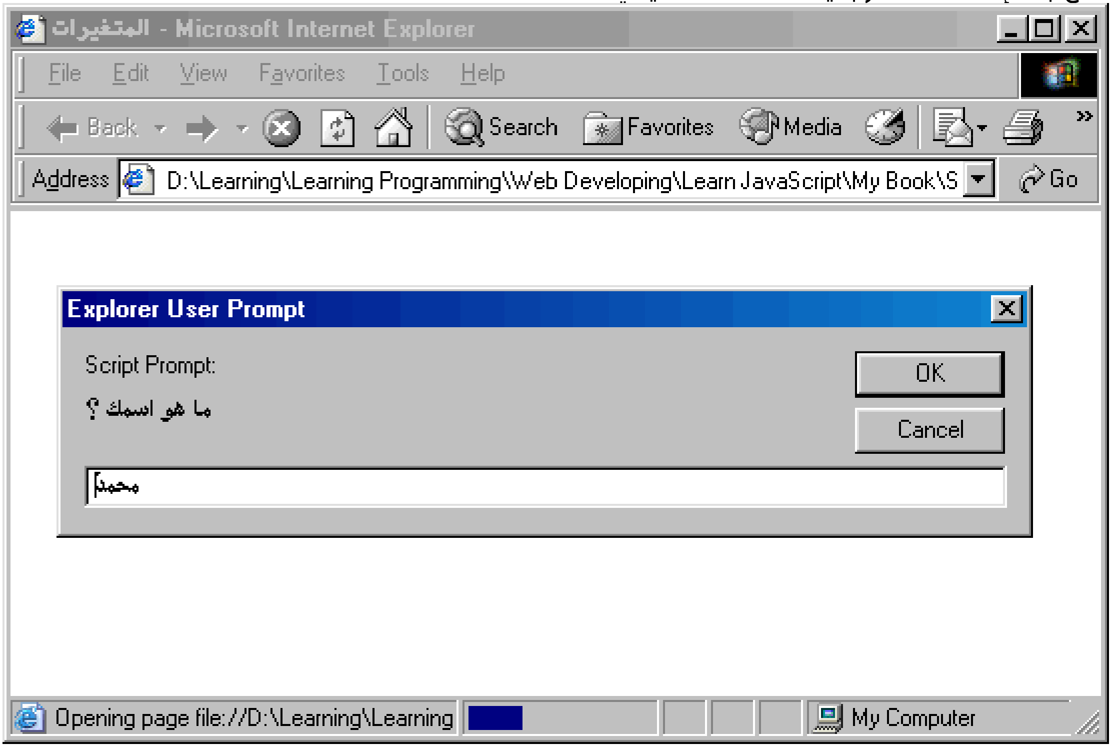

        alert( "مرحبا بك يا : " + username );

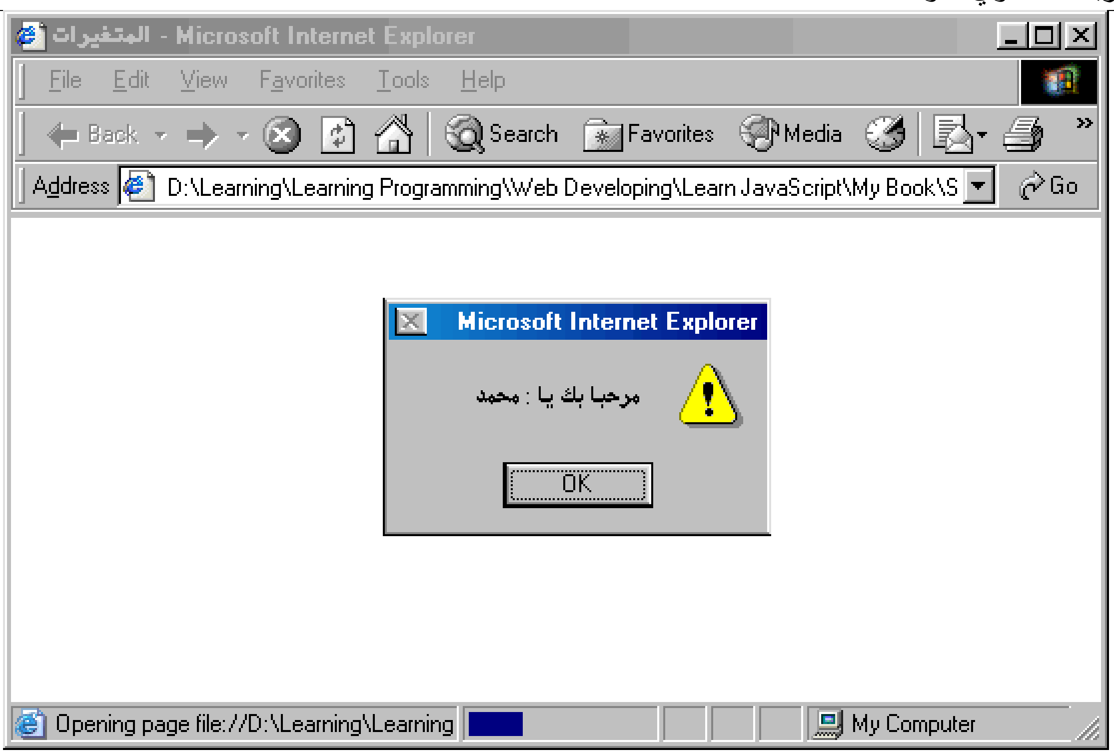
        document.write( "<b>مرحبا بك يا : </b>" + username + "<br>" );
        document.write( "<b>عمرك أقل من ٢٠ عام : </b>" + userage + "<br>" );
    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

ويكون الناتج بعد إختيار ok كما يلي :



ويكون الناتج بعد إدخال الأسم بقيمة **محمد** كما يلي :





المتغيرات - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Search Favorites Media

Address D:\Learning\Learning Programming\Web Developing\Learn JavaScript\My Book\S Go

مرحبا بك يا : محمد

عمرك أقل من ٢٠ عام : true

Done My Computer

الكلمات المحجوزة

أو مصطلحات الجافا سكربت المحجوزة ، وهي أوامر الجافا سكربت التي لايمكنك نسب أي متغيرات إليها على الإطلاق ، كما في كل لغات البرمجة المعروفة .

طبعا لايجب عليك حفظها كاملة ، وإنما الإلمام بها . وستجد جدولا مبينا لها

break	in	true
continue	int	typeof
do	labeled	var
else	new	void
false	null	while
for	return	with
function	switch	
if	this	

abstract	final	protected
boolean	finally	public
byte	float	short
case	goto	static
catch	implements	synchronized
char	import	super
class	instanceof	throw
const	interface	throws
default	long	transient
delete	native	try
double	package	
extends	private	

الفصل الرابع

المعاملات

سوف نناقش إن شاء الله في هذا الفصل النقاط التالية :

- المعاملات **Operators**
  - معاملات رياضية Arithmetic Operators
  - معاملات دمج النصوص Concatenation Operators
  - معاملات المقارنة Comparison Operators
  - معاملات منطقية Logical Operators
  - معاملات احادية Unary Operators
  - معاملات تغير القيم Assignment Operators
  - معاملات التعامل بالنظام الثنائي Shift & Bitwise Operators
  - معاملات خاصة

- أولوية تنفيذ المعاملات **Operator Precedence**
- معالجة الأخطأ **Exception Handling**
- جملة **with**

المعاملات Operators

تعتبر المعاملات هي الوسيط الأساسي للتعامل بين قيم المتغيرات .
وتنقسم المعاملات إلي غدة أقسام كما يلي :

- معاملات رياضية Arithmetic Operators
- معاملات دمج النصوص Concatenation Operators
- معاملات المقارنة Comparison Operators
- معاملات منطقية Logical Operators
- معاملات احادية Unary Operators
- معاملات تغير القيم Assignment Operators
- معاملات التعامل بالنظام الثنائي Shift & Bitwise Operators
- معاملات خاصة

معاملات رياضية Arithmetic Operators

تستخدم لإجراء العمليات الرياضية من جمع وطرح وضرب وقسمة وباقي القسمة .

- + : تستخدم لإجراء عمليات الجمع الرياضي .
- - : تستخدم لإجراء عمليات الطرح الرياضي .
- * : تستخدم لإجراء عمليات الضرب الرياضي .
- / : تستخدم لإجراء عمليات القسمة الرياضي .
- % : تستخدم لحساب باقي القسمة الرياضي

مثال :

```
alert( 5 % 2 );
```

ويكون الناتج كما يلي :



معاملات دمج النصوص Concatenation Operators

- + : كما تستخدم لإجراء عمليات الجمع الرياضي تستخدم لدمج النصوص كما يلي :

```
var username = "محمد";
alert( " : مرحبا بك يا" + username );
```

ويكون الناتج كما يلي :



معاملات المقارنة Comparison Operators

تستخدم لإجراء عمليات المقارنة
عند إجراء مقارنة يكون ناتج هذة المقارنة إحدي القيم اما true أو false .

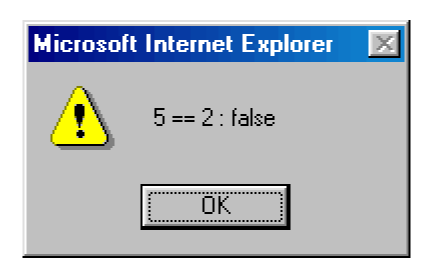
- == : لتعين هل طرفي المقارنة متساويان في القيمة .
- === : لتعين هل طرفي المقارنة متساويان في القيمة و نوع البيانات.
- != : لتعين هل طرفي المقارنة غير متساويان في القيمة .
- !== : لتعين هل طرفي المقارنة غير متساويان في القيمة و نوع البيانات.
- > : لتعين هل الطرف الأيسر أكبر من الطرف الأيمن .
- < : لتعين هل الطرف الأيسر أقل من الطرف الأيمن .
- >= : لتعين هل الطرف الأيسر أكبر أو يساوي الطرف الأيمن .
- <= : لتعين هل الطرف الأيسر أقل من أو يساوي الطرف الأيمن .

لاحظ :
عند تعين هل الطرفين متساويان يم أستخدام الحرف "=" مرتيين متتاليتين كما بالمثال التالي :

مثال :

```
alert( "5 == 2 : " + (5 == 2) );
alert( "7 == 7 : " + (7 == 7) );
```

ويكون الناتج كما يلي :





معاملات منطقية Logical Operators

تستخدم لإجراء مقارنة منطقية ، وغالبا ما تكون أطراف المقارنة إحدي القيم true أو false .
عند إجراء مقارنة يكون ناتج هذة المقارنة إحدي القيم اما true أو false .

- && :

تعني العملية المنطقية "**و**" وتقوم بإرجاع القيمة true عندما يكون طرفي المقارنة يساوي true وغير ذلك سوف يؤدي إلي إرجاع القيمة false كما يلي :

القيمة	القيمة	الناتج
True	True	True
True	False	False
False	True	False
False	False	False

- || :

تعني العملية المنطقية "**أو**" وتقوم بإرجاع القيمة true عندما يكون احدي طرفي المقارنة أو كلاهما يساوي true وغير ذلك سوف يؤدي إلي إرجاع القيمة false كما يلي :

القيمة	القيمة	الناتج
True	True	True
True	False	True
False	True	True
False	False	False

- ! :

تعني العملية المنطقية "**لا**" وتقوم بعكس القيمة كما يلي :

القيمة	الناتج
True	False
False	True

أمثلة :

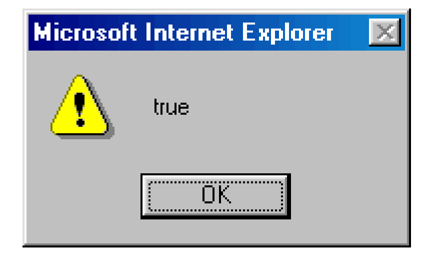
```
alert( true && false );
```

ويكون الناتج كما يلي :




```
alert( true || false );
```

ويكون الناتج كما يلي :



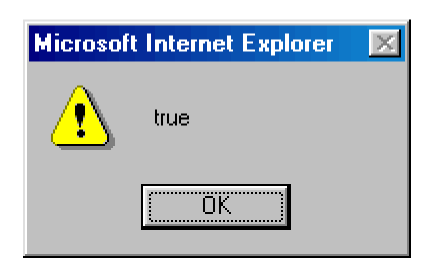

```
alert( !true );
```

ويكون الناتج كما يلي :



```
        alert( !( 5 > 7 ) );
```

ويكون الناتج كما يلي :



المعاملات الاحادية Unary Operators

تستخدم لإجراء العمليات الرياضية من جمع وطرح بشكل مختصر جدا .
وتسمي هذه المعاملات بأنها معاملات احادية لأنها تتعامل مع طرف واحد (operand)

- ++ : تستخدم لزيادة قيمة الطرف (العامل) الممرر لها بمقدار واحد صحيح .
- - - : تستخدم لإنقاص قيمة الطرف (العامل) الممرر لها بمقدار واحد صحيح .
- - : تستخدم لعكس إشارة العامل الممرر لها فإذا كانت قيمة سالبة فسوف تتحول إلي قيمة موجبة والعكس صحيح .

لاحظ :
يوجد حالتان للمعاملين ++ و -- حالة قبلية وأخري بعدية كما يلي :

الحالة البعدية (Postfix):
فيها يتم إجراء عملية الزيادة (++) أو النقص (--) بعد الإنتهاء من إجراء السطر الحالي لاحظ معي المثال التالي :

مثال :

```
<HTML dir=rtl>
  <Title> المعاملات الاحادية </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
           var num1 = 5, num2;

           num2 = num1++;

           alert( "num2 = " + num2 );
           alert( "num1 = " + num1 );
    //-->
```

```
    </SCRIPT>
  </HEAD>
</HTML>
```

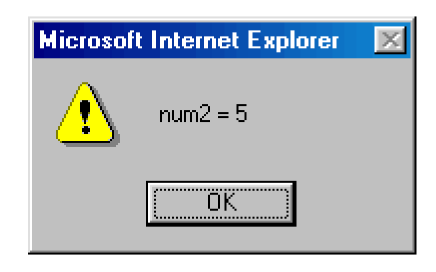
لاحظ السطر التالي :

```
           num2 = num1++;
```

يتم فيه استخدام المعامل الاحادي ++ ولكن بالوضع البعدي لأنه أتي بعد العامل num1
وهذا السطر يكافئ السطرين التتاليين :

```
           num2 = num1;
           num1 = num1 + 1;
```

لذلك يكون الناتج كما يلي :





الحالة القبلية (Prefix):
فيها يتم إجراء عملية الزيادة (++) أو النقص (--) قبل إجراء تنفيذ السطر الحالي لاحظ معي المثال التالي :

مثال :

```
<HTML dir=rtl>
  <Title> المعاملات الاحادية </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
           var num1 = 5, num2;
```

```
            num2 = ++num1;

            alert( "num2 = " + num2 );
            alert( "num1 = " + num1 );
    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

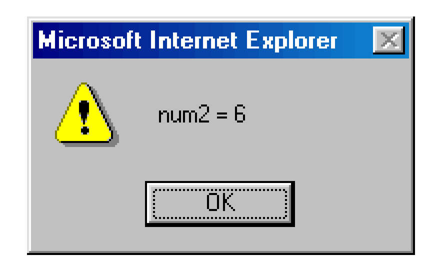
لاحظ السطر التالي :

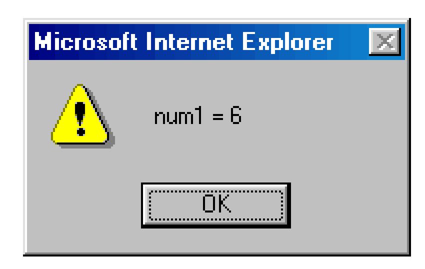
```
            num2 = ++num1;
```

يتم فيه استخدام المعامل الاحادي ++ ولكن بالوضع القبلي لأنه أتي قبل العامل num1
وهذا السطر يكافئ السطرين التتاليين :

```
            num1 = num1 + 1;
            num2 = num1;
```

لذلك يكون الناتج كما يلي :





لاحظ معي الأمثلة التالية :

مثال :

```
var num1 = 3, num2;
num2 = ++num1 + 4;

alert( "num1 = " + num1 );
alert( "num2 = " + num2 );
```

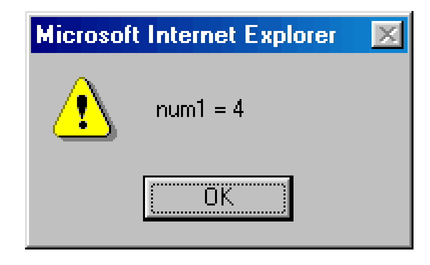
لا حظ ان السطر التالي :

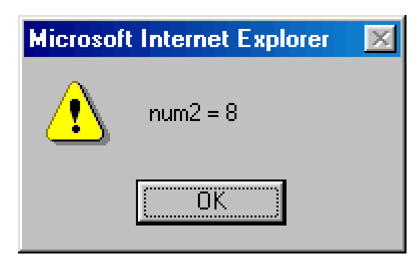
```
num2 = ++num1 + 4;
```

يكافي السطريين التتاليين :

```
num1 = num1 + 1;
num2 = num1 + 4;
```

لذلك يكون الناتج كما يلي :



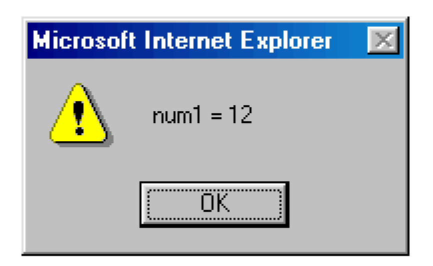


مثال :

```
var num1 = 13;
num1--;

alert( "num1 = " + num1 );
```

ويكون الناتج كما يلي :



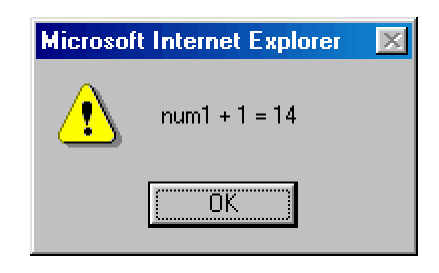
مثال :

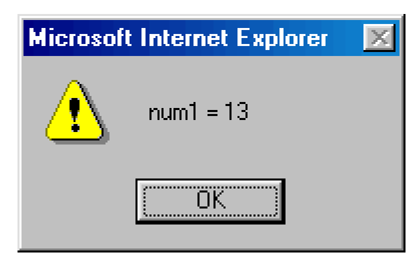
```
var num1 = 13;

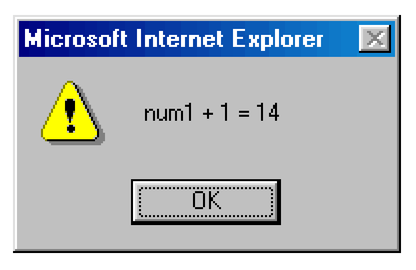
alert( "num1 + 1 = " + (num1 + 1) );
alert( "num1 = " + num1 );

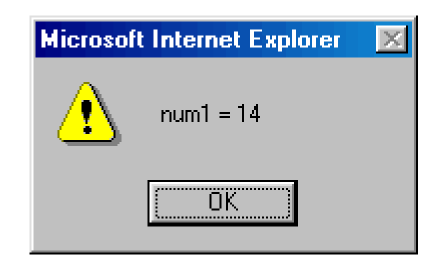
alert( "num1 + 1 = " + ++num1 );
alert( "num1 = " + num1 );
```

ويكون الناتج كما يلي :









لاحظ أن السطر التالي لم يؤثر في قيمة num1

```
alert( "num1 + 1 = " + (num1 + 1) );
```

معاملات تغير القيم Assignment Operators

تستخدم لتغير قيمة المعامل الموجود بالطرف الأيسر

- = : لتغير قيمة الطرف الأيسر بقيمة الطرف الأيمن .
- += : لزيادة قيمة الطرف الأيسر بمقدار قيمة الطرف الأيمن .
- -= : لنقص قيمة الطرف الأيسر بمقدار قيمة الطرف الأيمن .
- *= : لضرب قيمة الطرف الأيسر في قيمة الطرف الأيمن ووضع الناتج ليكون قيمة الطرف الأيسر .
- /= : لقسمة قيمة الطرف الأيسر علي قيمة الطرف الأيمن ووضع الناتج ليكون قيمة الطرف الأيسر .
- %= : لحساب باقي قسمة قيمة الطرف الأيسر علي قيمة الطرف الأيمن ووضع الناتج ليكون قيمة الطرف الأيسر .

مثال :

```
var num = 6;

num += 5;
alert( "num : " + num );
```

ويكون الناتج كما يلي :



لاحظ السطر التالي :

```
num += 5;
```

ويعني أضف إلي قيمة **المتغير** num المقدار 5 وهذا السطر يكافئ السطر التالي :

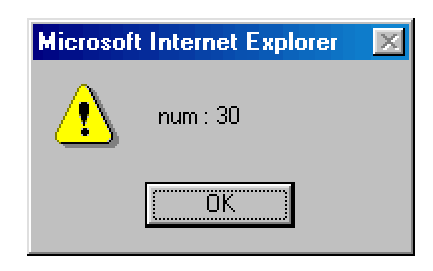
```
num = num + 5;
```

مثال :

```
var num = 6;

num *= 5;
alert( "num : " + num );
```

ويكون الناتج كما يلي :



معاملات التعامل بالنظام الثنائي Shift & Bitwise Operators

قبل الحديث عن معاملات النظام الثنائي دعني أخذك لنري لمحات في عالم الثنائيات أو نظام العد الثنائي ولا أخفي علي اي مبتدأ في عالم البرمجيات أو ربما المتوسطين أيضا أن دخول هذا المنعطف ربما لا يكون سهل ولكن دعنا نتوكل علي الله فهو خيرا معين .

النظام الثنائي

هو أحد أنظمة العد مثل النظام العشري المعتاد لنا .
لكن النظام الثنائي يعتمد علي رقمين فقط وهما **الواحد** و **الصفر** .
وهذا النظام العددي تم إستخدامه لبرمجة الحواسب في بدايات تطوير برامج للحاسبات بما كان يعرف بلغة الألة machine language وذلك لان وحدات بناء الحاسب تتعامل في حساباتها بالنظام الثنائي (لان الوحدة الإلكترونية لا تستطيع تميز إلا حالتان وهما وجود إشارة إليكترونية نتيجة إغلاق الدائرة الإلكترونية ، وعدم وجود إشارة نتيجة فتح الدائرة الإلكترونية) .

فإن كنا نريد تمثيل الأعداد ولا نملك إلا رقمين فقط وهما ١ وال ٠
لذلك قياسا علي ما تعلمناه في النظام العشري عندما نريد تمثيل العدد **عشرة** نقوم بعملية دمج الرقمين ١ و ٠
وعند تمثيل العدد **الحادي عشر** نقوم بدمج الرقمين ١ و ١ وهكذا عندما نخرج عن نطاق نظام العد .

وبما أن النظام الثنائي لا يتميز إلا برقمين فقط فعندما نريد تمثيل العدد **أثنين** نقوم بدمج الرقمين ١ و ٠ وهكذا
فالعدد العشري **٢** يمثل في النظام الثنائي كما يلي **١٠**
والعدد العشري **٣** يمثل في النظام الثنائي كما يلي **١١**

وتوجد أساليب رياضية لن نتطرق لها تستخدم في التحويل بين أنظمة العد المختلفه .

معاملات التعامل بالبت Bitwise Operators
تستخدم لتعامل مع القيمة الثنائية لكلا الطرفين
وعمل مقارنه منطقية لكل بت علي حدي كما سيتضح فيما يلي :

- & :
تتعامل مثل المعامل المنطقي &&
ولكن تتعامل ليس بالقيمة الكلية للطرفين ولكن تتم المقارنة علي مستوي البت ، بت واحد تلو الأخر .
- | :
تتعامل مثل المعامل المنطقي ||
ولكن تتعامل ليس بالقيمة الكلية للطرفين ولكن تتم المقارنة علي مستوي البت ، بت واحد تلو الأخر .
- ~ :
تتعامل مثل المعامل المنطقي !
ولكن تتعامل ليس بالقيمة الكلية للطرفين ولكن تتم المقارنة علي مستوي البت ، بت واحد تلو الأخر .
- ^ :
تتعامل بمفهوم الـ XOR مثل المعامل المنطقي || ولكن
تقوم بإرجاع القيمة true عندما يكون احدي طرفي المقارنة يساوي true وغير ذلك سوف يؤدي إلي إرجاع القيمة false كما يلي :

الناتج	القيمة	القيمة
False	True	True
True	False	True
True	True	False
False	False	False

مثال :

```
alert( 2 & 3 );
```

ويكون الناتج كما يلي :



تفسير الناتج كما يلي :

٠	١	٢
١	١	٣
٠	١	٢

مثال :

```
alert( 2 | 3 );
```

ويكون الناتج كما يلي :



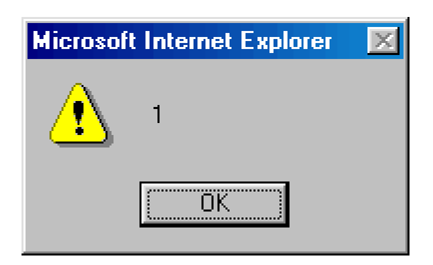
تفسير الناتج كما يلي :

٠	١	٢
١	١	٣
١	١	٣

مثال :

```
alert( 2 ^ 3 );
```

ويكون الناتج كما يلي :



تفسير الناتج كما يلي :

٠	١	٢
١	١	٣
١	٠	١

أما **معاملات الإزاحة** Shift Operators

تستخدم لعمل أزاحة (تحريك) للبتات bits إلي اليمين أو اليسار حسب نوع الإزاحة كما يلي

- > > :
  تقوم بعمل إزاحة ناحية اليمين لعدد من البتات يكافيء مقادر قيمة الطرف الأيمن لقيمة العدد بالطرف الأيسر
- < < :
  تقوم بعمل إزاحة ناحية اليسار لعدد من البتات يكافيء مقادر قيمة الطرف الأيمن لقيمة العدد بالطرف الأيسر

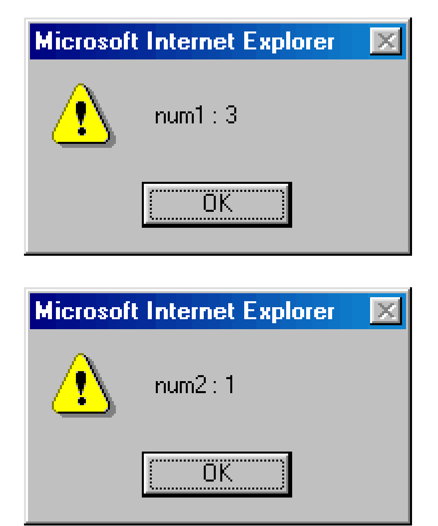
مثال :

```
var num = 6 , num1 , num2;

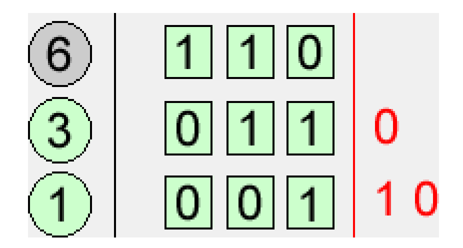
num1 = num >> 1;
alert( "num1 : " + num1 );

num2 = num >> 2;
alert( "num2 : " + num2 );
```

ويكون الناتج كما يلي :



وتفسير الناتج كما يلي :
يمثل العدد ٦ بالعدد ١١٠ بالنظام الثنائي ، فعند إزاحة هذا العدد الثنائي ناحية اليمين بمقدار بت واحد سوف يصبح كالتالي ١١ اي ما يكافئ العدد ٣ بالنظام العشري ، وإذا تم إزاحة بمقدار بت أخر جهة اليمين يكون العدد كالتالي ١ اي ما يعادل ١ بالنظام العشري كما يتضح بالشكل التالي :



مثال :

```
var num = 6 , num1 , num2;

num1 = num << 1;
alert( "num1 : " + num1 );
```

```
        num2 = num << 2;
        alert( "num2 : " + num2 );
```

ويكون الناتج كما يلي :





وتفسير الناتج كما يلي :
يمثل العدد **٦** بالعدد **١١٠** بالنظام الثنائي ، فعند إزاحة هذا العدد الثنائي ناحية اليسار بمقدار بت واحد سوف يصبح كالتالي **١١٠٠** اي ما يكافئ العدد **١٢** بالنظام العشري ، وإذا تم إزاحة بمقدار بت أخر جهة اليمين يكون العدد كالتالي **١١٠٠٠** اي ما يعادل **٢٤** بالنظام العشري

معاملات خاصة

- : ? :
  المعامل الشرطي (conditional operator) ويسمي أيضا معامل تيرنري (Ternary operator)
  الشكل العام له :

```
( الشرط )? القيمة عندما يكون ناتج الشرط صحيحة : القيمة عندما يكون ناتج الشرط خطأ ;
```

مثال :

```
var num1 = 6 ,num2;

num2 = ( num1 > 3 )? 5 : 100;
alert( "num2 : " + num2 );
```

ولأن ناتج المقارنة (num1 > 3) سوف يكون true
لذلك يكون الناتج كما يلي :

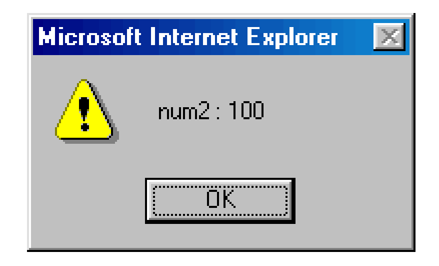


مثال :

```
var num1 = 6 ,num2;

num2 = ( num1 < 3 )? 5 : 100;
alert( "num2 : " + num2 );
```

ولأن ناتج المقارنة (num1 < 3) سوف يكون false
لذلك يكون الناتج كما يلي :



أولوية تنفيذ المعاملات Operator Precedence

كما تعلمنا جميعا بالمرحلة الإبتدائية أن العمليات الرياضية من جمع و طرح وقسمة وضرب تختلف أولوية تنفيذه حيث أن أولوية إجراء عمليات الضرب والقسمة أكبر من أولوية إجراء عمليات الجمع و الطرح .
وأولوية عمليات الضرب والقسمة متساوية .
وأيضا أولوية عمليات الجمع والطرح متساوية .
فجزي الله كل خير من علمنا العلم ونحن صغارا (وتالله ما زلنا صغارا ولسنا بصغار) .

تعالي معي نبحث هذه العملية الحسابية

```
2 + 4 * 3
```

فما نتيجة هذه العملية الحسابية هل يكون الناتج **١٨** أم **١٤** ؟
لو تم جمع الرقم ٢ علي الرقم ٤ ينتج العدد ٦ ثم عند ضرب هذا الرقم في العدد ٣ يكون الناتج ١٨ ولكن هذا غير صحيح رياضيا لأننا أجرينا عملية الجمع أولا ثم إجرينا عملية الضرب ، ولكننا علمنا من قليل أن أولوية إجراء عمليات الضرب والقسمة أكبر من أولوية إجراء عمليات الجمع و الطرح ، أي أننا ينبغي أن ننفذ عملية الضرب أولا وتكون الخطوات كالتالي :
يتم ضرب العدد ٤ في العدد ٣ وينتج عنه العدد ١٢ ثم يتم جمعه علي العدد ٢ ويكون الناتج ١٤ .

مثال :

```
var num1 = 2 ,num2 = 4 , num3;

num3 = num1 + num2 * 3;
alert( "num3 : " + num3 );
```

و يكون الناتج كما يلي :



ولكننا تعلمنا مما درسناه في منهج الرياضيات أننا يمكننا تغير أولوية تنفيذ العمليات الرياضية بإستخدام ما يسمى **أقواس المجموعات** وتكون العمليات الحسابية الموجودة داخل هذه الأقواس ذات أعلي أولوية كما بالمثال التالي :

مثال :

```
var num1 = 2 ,num2 = 4 , num3;

num3 = (num1 + num2) * 3;
alert( "num3 : " + num3 );
```

و يكون الناتج كما يلي :



لاحظ معي السطر التالي :

```
num3 = (num1 + num2) * 3;
```

لاحظ وجود الأقواس حول العملية الحسابية num1 + num2
مما يجعل أولوية إجراء هذه العملية أكبر من أولوية إجراء عملية الضرب .
فيتم جمع الرقم ٢ علي الرقم ٤ ينتج العدد ٦ ثم عند ضرب هذا الرقم في العدد ٣ يكون الناتج ١٨

مثال :

```
var num1 = 2 ,num2 = 4 , num3;

num3 = ( (num1 + num2) / 2 + 6 )  * 3;
alert( "num3 : " + num3 );
```

و يكون الناتج كما يلي :



معالجة الأخطا Exception Handling

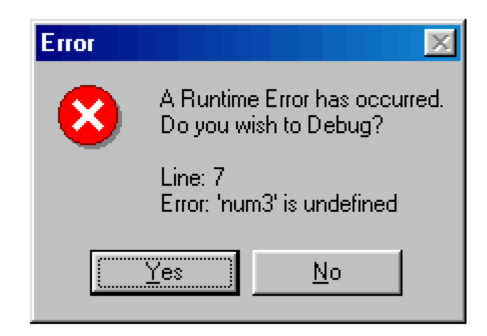
ربما يصادفك حدوث بعض المشاكل ولا تكون أخطاء لغوية ومع أنك لم تخطأ في كود البرنامج دعنا نري المثال التالي

مثال :

```
<HTML dir=rtl>
  <Title> معالجة الأخطا </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
           var num1 ,num2;

           alert( num3 );
    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

وبما أن المتغير المسمى num3 لم يتم تعريفة ولا إعطائه قيمة فلا يمكن إستخدامه
ويكون الناتج كما يلي :



كما يتضح من الرسالة أنها رسالة Error اي رسالة خطأ
وتشير الرسالة إلي أن "num3 is undefined" اي أن المتغير المسمى num3 غير معرف أو غير معروف لدى البرنامج .

ولكن السؤال الأن هل يمكن تجنب وقوع الأخطأ ؟
نعم ، يمكنك تجنب حدوث الأخطأ باستخدام صيغة try و الـ catch

الصيغة العامة

```
try{

}catch( e ){

}
```

يتم كتابة الأكود بين أقواس الجملة try فإذا حدث اي خطأ ينتقل المفسر إلي منطقة الأكواد بين أقواس الجملة catch ويتم تحميل كائن مخصص لحمل مواصفات هذا الخطأ جرت العادة لتسمية هذا المتغير باسم e لانه يدل علي حدوث إستثناء Exception

ولمعالجة المثال السابق نقوم بإعادة كتابة كالتالي
مثال :

```
<HTML dir=rtl>
  <Title> معالجة الأخطا </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
           var num1 ,num2;
           try{
              alert( num3 );
           }catch( e ){
              alert("حدث خطأ");
           }
    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

ويكون الناتج كما يلي



ويمكننا إظهار رسالة الخطأ الأساسية بإستخدام خصائص الكائن e وخائصه هي :

- **Name**
  وتعبر عن اسم الخطأ
- **Message**
  وتعبر عن محتوى نص رسالة الخطأ
- **Number**
  وتعبر عن رقم الخطأ
- **Description**
  وتعبر عن رسالة تفصيلية للخطأ

نقوم بإعادة كتابة المثال السابق كالتالي
مثال :

```
<HTML dir=rtl>
  <Title> معالجة الأخطاء </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
           var num1 ,num2;
           try{
              alert( num3 );
           }catch( e ){
              alert( "اسم الخطأ : " + e.name + "\n" +
                     "رقم الخطأ : " + e.number + "\n" +
                     "رسالة  الخطأ : " + e.message + "\n" +
                     "وصف الخطأ : " + e.description );
           }
    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

ويكون الناتج كما يلي



إستحداث خطأ بإستخدام الجملة throw

في بعض الأحيان نحتاج إلي إستحداث خطأ ما ، ويتم ذلك بإستخدام الأمر throw ، بطريقتين

أولهما
هذا المثال ليس به اي خطأ ولكننا سوف نستحدث هذا الخطأ كما يلي

```
<HTML dir=rtl>
  <Title> معالجة الأخطا </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
           var helloMessage = "مرحبا بك";
           try{
              alert( helloMessage );
              throw "هذا خطأ ليس حقيقي ولكنه مستحدث";
           }catch( e ){
              alert( " : رسالة الخطأ" + e );
           }
    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

ويكون الناتج كما يلي



من المثال السابق تم كتابة رسالة الخطأ بعد الأمر throw كما بالسطر التالي

```
              throw "هذا خطأ ليس حقيقي ولكنه مستحدث";
```

فعند تنفيذ هذا الأمر ينتقل المفسر إلي الجملة catch حاملا معه رسالة الخطأ في المتغير e
بعدها قمنا بطباعة رسالة الخطأ كما يلي

```
           }catch( e ){
              alert( " : رسالة الخطأ" + e );
           }
```

ثانيهما هو عمل خطأ من الكائن Error ثم إرساله إلي الأمر throw
كما بالمثال التالي

مثال

```
<HTML dir=rtl>
  <Title> معالجة الأخطا </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
           var helloMessage = "مرحبا بك";
           try{
              alert( helloMessage );
              var err = new Error("هذا خطأ ليس حقيقي ولكنه مستحدث");
              err.description = "هذا خطأ تم صنعة يدوين";
              throw err;
           }catch( e ){
              alert( "رسالة الخطأ : " + e.message + "\n" +
                     "وصف الخطأ : " + e.description );
           }
    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

ولن نخوض كثيرا في هذا المثال حتي نقوم بشرح الكائن Error بالجزء الثاني من هذا الكتاب إن شاء الله

جملة with

عندما تقوم بإستخدام خصائص properties و أو وظائف functions تابعة لكائن واحد ، يمكننا أن نجعل هذا الكائن يشير إلي خصائصه و وظائفه بدون تكرار كتابته قبل كتابة الخاصية أو الوظيفة .

الصيغة العامة لجملة with

```
    with( اسم الكائن المستهدف عدم تكراره ) {
            // الأكواد
            // ربما تحتوي الأكواد علي خصائص أو وظائف لهذا الكائن
    }
```

وبالمثال يتضح المقال

مثال :

هذا المثال بدون إستخدام جملة with

```
<HTML dir=rtl>
  <Title> بدون جملة with </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
            var userName = "الحسين محمد علي";

            var familyName = window.prompt("ما هو اسم العائلة ؟");
            window.alert(" : مرحبا بك يا" + userName );
            window.alert(" : مرحبا بعائلة" + familyName );

    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

وكما تري بالمثال السابق قد قمنا بتكرار كتابة الكائن window كما يلي

```
            var familyName = window.prompt("ما هو اسم العائلة ؟");
            window.alert(" : مرحبا بك يا" + userName );
            window.alert(" : مرحبا بعائلة" + familyName );
```

ويمكننا أن نقوم بإلغاء تكرار كتابة الكائن window بإستخدام الكائن window كما بالمثال التالي :

هذا المثال بإستخدام جملة with

```
<HTML dir=rtl>
  <Title> جملة with </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
           var userName = "الحسين محمد علي";

           with( window ){
              var familyName = prompt("ما هو اسم العائلة ؟");
              alert(" : مرحبا بك يا" + userName );
              alert(" : مرحبا بعائلة" + familyName );
           }
     //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

ويؤدي هذا المثال إلي نفس الناتج بالمثال السابق

ويتضح طريقة إستخدام جملة with بالأسطر التالية :

```
           with( window ){
              var familyName = prompt("ما هو اسم العائلة ؟");
              alert(" : مرحبا بك يا" + userName );
              alert(" : مرحبا بعائلة" + familyName );
           }
```

كما تري لم نقم بتكرار كتابة اسم الكائن window ، بل تم كتابته مرة واحدة بين أقواس جملة with ثم اي خاصية أو وظيفة تابعة لهذا الكائن تم كتابتها بين أقواس المجموعة لجملة with يمكنك تجاهل كتابة اسم الكائن window التابع له .

ملاحظة :
يمكنك كتابة اسم الكائن وحتي بعد إستخدام جملة with ولكن هذا يجعل جملة with بدون فائدة .
كما يلي :

```
           with( window ){
              var familyName = prompt("ما هو اسم العائلة ؟");
              window.alert(" : مرحبا بك يا" + userName );
              alert(" : مرحبا بعائلة" + familyName );
           }
```

تداخل جملة with :

يسمح بعمل تداخل nesting لجمل with كما بالمثال التالي :

مثال

```
<HTML dir=rtl>
  <Title> جملة with </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
           var userName = "الحسين محمد علي";

           with( window ){
              var familyName = prompt("ما هو اسم العائلة ؟");
              alert(" : مرحبا بك يا" + userName );
              alert(" : مرحبا بعائلة" + familyName );

              //  تداخل لجمل with
              with( document ){
                     write(" مرحبا بفكرة تداخل جمل with <br>" );
                     writeln("  إلي اللقاء مع الفصل القادم ");
                     alert(" : إلي اللقاء لعائلة" + familyName );
              }
           }
     //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

كما تلاحظ أنه تم تداخل جملتين لـ with أحدهما للكائن window والأخري للكائن document كما يلي :

```
           with( window ){
              var familyName = prompt("ما هو اسم العائلة ؟");
              alert(" : مرحبا بك يا" + userName );
              alert(" : مرحبا بعائلة" + familyName );

              //  تداخل لجمل with
              with( document ){
                     write(" مرحبا بفكرة تداخل جمل with <br>" );
                     writeln("  إلي اللقاء مع الفصل القادم ");

                     //  لاحظ أن هذه الوظيفة alert
                     //  تابعة لجملة
                     // with  الأولي
                     alert(" : إلي اللقاء لعائلة" + familyName );
              }
           }
```

الفصل الخامس

التحكم في مسار البرنامج

سوف نناقش إن شاء الله في هذا الفصل النقاط التالية :

- جمل الشرط **Conditional Statements**
  - جملة الشرط **if**
  - جملة الشرط switch
- جمل التكرار والحلقات **Iteration Statements**
  - جملة التكرار for
  - جملة التكرار while
  - جملة التكرار do while

جمل الشرط Conditional Statements

دائما في حياتنا المعاصرة ما نحتاج لإتخاذ بعض القرارات هل نفعل كذا أم كذا علي سبيل المثال قبل أن تبدأ في تعلم لغة الجافا سكربت ربما تبادر لك هذا السؤال **هل أنت في حاجة لتعلم هذه اللغة ؟**
وهذا السؤال يحتمل إحدي الإجابات إما نعم أو لا ، وبإجابتك علي هذا السؤال تكون قد حددت مسارك في الأيام القادمة ، علي سبيل المثال لو كانت إجابتك **بلا** لما إستطعت قرأت هذا النص المكتوب بين يديك ، وإذا كانت إجابتك بنعم فسوف تكون من قارئي هذا النص .

نخلص مما سبق إلي أننا دائما ما تطرح أمامنا **إختيارات** نحتاج لوضعها في **جمل شرط** وهذه الجملة تحدد المسار حسب الإختيار ، فعلي سبيل المثال لجملة الشرط
هل تريد أن تتعلم لغة الجافا سكربت ؟ وليس أمامك إلا إختياران أحدهما **نعم** والأخر **لا**
فعندما تقول نعم فسوف تجني ثمار ما تعلمت إن شاء الله فمن جد وجد ، وإذا كان الخيار بلا فلا تلومنا إلا نفسك لتفريطك في أخذ العلم النافع .

جملة الشرط if

من أشهر جمل الشرط الجملة if وتستخدم لعمل تفريع بمسار البرنامج .

الصيغة العامة لجملة الشرط if
توجد عدة صيغ لجملة الشرط if كما يلي :

الصيغة الفردية if :
وفيه يتم حساب قيمة **جملة الشرط** فإذا كانت جملة الشرط تكافي القيمة true (أو اي قيمة عددية غير الصفر أو اي قيمة نصية غير قيمة النص الفارغ أو القيمة null) ، فعندئذ يتم تنفيذ الأكواد بين أقواس جملة الشرط if ، وإذا كانت تكافئ القيمة false (أو القيمة العددية صفر أو القيمة النصية الفارغة) سوف يتم تجاهل تنفيذ الأكواد بين أقواس جملة الشرط .

```
if( جملة الشرط ) {
        //  الأكواد
}
```

مثال

```
var num = 78;
if( num >= 50  ) {
        alert( " المتغير num = " + num + "\n ٥٠ أكبر من أو يساوي "   );
}
```

ويكون الناتج كما يلي



الصيغة المزدوجة if ... else :

وفيه يتم حساب قيمة **جملة الشرط** فإذا كانت جملة الشرط تكافي القيمة true (أو اي قيمة عددية غير الصفر أو اي قيمة نصية غير قيمة النص الفارغ أو القيمة null) ، فعندئذ يتم تنفيذ الأكواد بين أقواس جملة الشرط if ، وإذا كانت تكافئ القيمة false (أو القيمة العددية صفر أو القيمة النصية الفارغة) سوف يتم تجاهل تنفيذ الأكواد بين أقواس جملة الشرط if ثم يتم تنفيذ الأكواد المنحصرة بين أقواس الجملة else .

```
if( جملة الشرط ) {
        // الأكواد
}else{
        // الأكواد
}
```

مثال

```
var num = 18;
if( num >= 50  ) {
        alert( " المتغير num = " + num + "\n ٥٠ أكبر من أو يساوي "   );
}else{
        alert( " المتغير num = " + num + "\n ٥٠ أقل من "   );
}
```

ويكون الناتج كما يلي



الصيغة المركبة :

وفيه يتم حساب قيمة **جملة الشرط الأولي** فإذا كانت جملة الشرط تكافي القيمة true (أو اي قيمة عددية غير الصفر أو اي قيمة نصية غير قيمة النص الفارغ أو القيمة null) ، فعندئذ يتم تنفيذ الأكواد بين أقواس جملة الشرط if ، وإذا كانت تكافئ القيمة false (أو القيمة العددية صفر أو القيمة النصية الفارغة) سوف يتم تجاهل تنفيذ الأكواد بين أقواس جملة الشرط if ثم ينتقل ألي جملة الشرط التالية ويفعل معها مثل ما سبق فإذا كانت كل جمل الشرط تؤول إلي false فسوف يتم البحث عن جملة else فإن وجدت فسوف يتم تنفيذ الأكواد المنحصرة بين أقواس الجملة else وإن لم يجد جملة else فلن يتم تنفيذ شئ.

```
if( جملة الشرط رقم ١ ) {
        // الأكواد
else if( جملة الشرط رقم ٢ ) {
        // الأكواد
}else{
        // الأكواد
}
```

أو بدون else كما يلي :

```
if( جملة الشرط رقم ١ ) {
        // الأكواد
else if( جملة الشرط رقم ٢ ) {
        // الأكواد
}
```

مثال

```
var num = 48;

if( num >= 1 && num <= 10 ) {
        alert( " العدد num \n محصور بين ١ و ١٠" );
}else if( num > 10  && num <=20 ){
        alert( " العدد num \n أكبر من ١٠ وأقل من أو تساوي ٢٠" );
}else if( num > 20  && num <=30 ){
        alert( " العدد num \n أكبر من ٢٠ وأقل من أو تساوي ٣٠" );
}else{
        alert( "العدد num \n أكبر من ٣٠" );
}
```

ويكون الناتج كما يلي



مثال
هو نفس المثال السابق ولكن بدون جملة else

```
var num = 48;

if( num >= 1 && num <= 10 ) {
        alert( " العدد num \n محصور بين ١ و ١٠" );
}else if( num > 10  && num <=20 ){
        alert( " العدد num \n أكبر من ١٠ وأقل من أو تساوي ٢٠" );
}else if( num > 20  && num <=30 ){
        alert( " العدد num \n أكبر من ٢٠ وأقل من أو تساوي ٣٠" );
}
```

ويكون الناتج كما يلي
لن يتم ظهور اي رسائل لعدم تحقق اي من الشروط السابقة .

تمرين للكشف عن كلمة السر

```
<HTML dir=rtl>
  <Title> الكشف عن كلمة السر </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
           var userPassword = "123";

           var password = prompt("أدخل كلمة السر", "");
           if( password == userPassword ){
              alert( "مرحبا بك يا \n كلمة سر صحيحة" );
           }else{
              alert( "كلمة السر غير صحيحة" );
           }
    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

ويكون الناتج بعد إدخال كلمة السر بقيمة ١٢ ثم الضغط علي زرار ok كما يلي :



ويكون الناتج بعد إدخال كلمة السر بقيمة ١٢٣ ثم الضغط علي زرار ok كما يلي :



حاول أن تركز في الأمثلة القادمة :

مثال

```
var num = 4;

if( ++num == 5 ) {
        alert( " العدد num \n ٥ يساوي" );
}
```

ويكون الناتج كما يلي

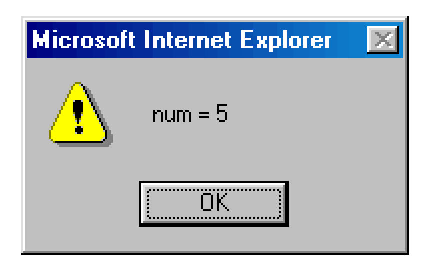


مثال

```
var num = 4;

if( num++ == 5 ) {
        alert( " العدد num \n ٥ يساوي" );
}else{
        alert( "num = " + num );
}
```

ويكون الناتج كما يلي



جملة الشرط switch

تستخدم جملة الشرط switch مثل إستخدامنا لجملة الشرط if المزدوجة أو المركبة ، إذا ما دامت تؤدي نفس وظيفة الجملة if فما الحاجة إليها ، نعم أنت علي حق لكننا نحتاج إلي جملة switch الشرطية بدلا من جملة else ... if لتسهيل شكل الكود .

الصيغة العامة

```
switch( المتغير أو العملية المراد مقارنتها ) {
        case "قيمة ١" :
           // الأكواد
           break;
        case "قيمة ٢" :
           // الأكواد
           break;
        case "قيمة ٣" :
           // الأكواد
           break;
        default :
           // الأكواد
}
```

أو بدون default كما يلي :

```
switch( المتغير أو العملية المراد مقارنتها ) {
        case "قيمة ١" :
           // الأكواد
           break;
        case "قيمة ٢" :
           // الأكواد
           break;
        case "قيمة ٣" :
           // الأكواد
           break;
}
```

يتم مقارنة **المتغير أو العملية المراد مقارنتها** بالقيم الموجودة بحالات (cases) جملة switch فإذا تساوت إحداهما مع قيمة المتغير المقارن يتم الدخول داخل هذة الحالة case وتنفيذ الأكواد التي بداخلها ، وإن لم يتم تساوي القيمة المقارنة باي من حالات جملة switch يتم البحث عن الحالة الخاصة default فإذا وجدت يتم الدخول فيها ثم تنفذ الأكواد الموجودة بها .

ملاحظة :
لتفادي حدوث **عملية السقوط** أو ما يسمي بي fall through يجب وضع الأمر break حتي تنهي جملة switch .
ولكن في بعض الأحيان نحتاج لعدم تلافي عملية السقوط كما سنري بالأمثلة القادمة .

مثال

```
<HTML dir=rtl>
  <Title>  جملة الشرط switch </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
    var num = 4;

    switch( num ) {
           case 1 :
              alert( " العدد num \n ١ يساوي" );
              break;
           case 2 :
              alert( " العدد num \n ٢ يساوي" );
              break;
           case 3 :
              alert( " العدد num \n ٣ يساوي" );
              break;
           case 4 :
              alert( " العدد num \n ٤ يساوي" );
              break;
           default :
              alert( " العدد num \n  يساوي " + num );
              break;
    }
    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

ويكون الناتج كما يلي



مثال

```
<HTML dir=rtl>
  <Title> جملة الشرط switch </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
    var num = 4;

    switch( num ) {
           case 1 , 2 , 3 , 4:
              alert( " العدد num \n ربما يساوي ١ أو ٢ أو ٣ أو ٤" );
              break;
           default :
              alert( " العدد num \n  يساوي " + num );
              break;
    }
    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

كما تلاحظ بالسطر ذو اللون الأحمر
يتم دخول هذه الحالة إذا تساوت القيمة num بإحدي القيم ١ أو ٢ أو ٣ أو ٤

```
    switch( num ) {
           case 1 , 2 , 3 , 4:
              alert( " العدد num \n ربما يساوي ١ أو ٢ أو ٣ أو ٤" );
              break;
```

ويكون الناتج كما يلي



لاحظ معي التالي :
أن هذه الحالة تكافيء التالي :

```
    switch( num ) {
           case 1 :
           case 2 :
           case 3 :
           case 4 :
              alert( " العدد num \n ربما يساوي ١ أو ٢ أو ٣ أو ٤" );
              break;
```

كما بالمثال التالي

```
<HTML dir=rtl>
  <Title> جملة الشرط switch </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
    var num = 4;

    switch( num ) {
           case 1 :
           case 2 :
           case 3 :
           case 4 :
              alert( " العدد num \n ربما يساوي ١ أو ٢ أو ٣ أو ٤" );
              break;
           default :
              alert( " العدد num \n  يساوي " + num );
              break;
    }
    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

وهنا لم نضع الجملة break لتفادي حدوث **عملية السقوط** fall through بل أردنا إحداث هذا السقوط ، و إذا تصادف تساوي القيمة num بإحدي القيم ١ أو ٢ أو ٣ أو ٤ سوف يتم تنفيذ نفس قطعة الكود .

فإذا كانت القيمة num تساوي القيمة ٢ ، سوف يتم السقوط إلي الحالة ٣ لأن الحالة ٢ لا يوجد بها الأمر break ، ثم يحدث سقوط أخر إلي الحالة ٤ لأن الحالة ٣ ليس بها الأمر break الخاص بإنهاء تنفيذ جملة switch ، ثم يتم تنفيذ الكود الموجود بالحالة ٤ ، ثم يتم إنهاء الجملة switch نتيجة وجود الأمر break بالحالة ٤ .

جمل التكرار والحلقات Iteration Statements

الهدف من جمل التكرار هو تكرار مجموعة من الأكواد لعدد أو مدي معين من المرات .

هب أنني طلبت منك أن تقوم بطباعة الجملة التالية "**أن محتاج لجمل التكرار**" ثلاث مرات فسوف تقول لي هذا أمر سهل ، نعم سوف أصدقك ولكن ماذا لو زادت مرات التكرار إلي ١٠٠٠ مرة مثلا أو إلي عدد من المرات غير محدد يتوقف علي شرط ما كما سنري عند الحديث علي جمل التكرار علي حدا .

جملة التكرار for

الهدف من جملة التكرار for هو تكرار مجموعة من الأكواد لعدد معين من المرات .

الصيغة العامة

```
for ( كود البداية ; شرط التكرار ; إجراء زيادة أو نقص ) {
        // الأكواد
}
```

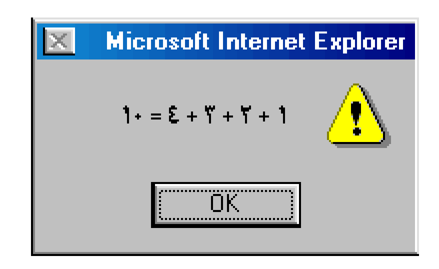
مثال
يقوم بطباعة مجموع الأرقام من ١ إلي ٤

```
<HTML dir=rtl>
  <Title>  جملة التكرار for</Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
    var x = 0;

    for( i=0; i<5 ; i++ ) {
           x += i;
    }

    alert("1 + 2 + 3 + 4 = " + x);
    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

ويكون الناتج كما يلي



عند تنفيذ جملة for يتم تنفيذ جملة البداية **أول مرة فقط** .

```
for( i=0; i<5 ; i++ ) {
        x += i;
}
```

ثم يتم حساب جملة الشرط فإذا كانت تساوي true يتم تنفيذ الأكواد بين أقواس الجملة for .

```
for( i=0; i<5 ; i++ ) {
        x += i;
}
```

ثم يتم تنفيذ الأكواد بين أقواس الجملة for .

```
for( i=0; i<5 ; i++ ) {
        x += i;
}
```

ثم يتم حساب جملة الزيادة أو النقص .

```
for( i=0; i<5 ; i++ ) {
        x += i;
}
```

ثم يتم حساب جملة الشرط فإذا كانت تساوي true يتم تنفيذ الأكواد بين أقواس الجملة for .

```
for( i=0; i<5 ; i++ ) {
        x += i;
}
```

وهكذا حتي تساوي جملة الشرط false فتنتهي هذة الحلقة .

مثال
مثل المثال السابق ولكن بعدم وضع جملة الشرط لجملة for

```
<HTML dir=rtl>
  <Title>   حملة التكرار for</Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
    var x = 0;

    for( i=0; ; i++ ) {
           x += i;
           if( i == 4 ) break;
    }

    alert("1 + 2 + 3 + 4 = " + x);
    //-->
```

```
    </SCRIPT>
  </HEAD>
</HTML>
```

ويكون الناتج كما بالمثال السابق

لاحظ أنه لا توجد جملة شرط لجملة for ولكن يوجد داخل كود جمل التكرار أحد جمل الشرط بها كود يؤدي إلي إنهاء دورة جملة for .

كما بالسطور التالية

```
    for( i=0; ; i++ ) {
           x += i;
           if( i == 4 ) break;
    }
```

جملة for in

تستخدم لعمل تكرار للمتغيرات من النوع الكائني objects مثل المصفوفات Arrays
وسوف نتحدث عنها في الجزء الثاني من هذا الكتاب .

الصيغة العامة

```
    for ( المتغير in الكائن) {
           // الأكواد
    }
```

جملة break

تستخدم لإنهاء جمل التكرار مثل جمل for وجمل while وجمل do while

جملة continue

تستخدم للرجوع مرة أخري إلي جملة الشرط المحددة لعملية التكرار ، وإلغاء تنفيذ الأكواد التالية لها كما بالمثال التالي :

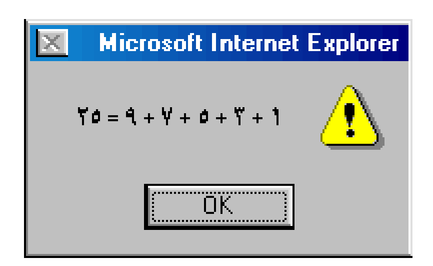
مثال
يقوم بطباعة مجموع الأرقام الفردية بين ١ و ١٠

```
<HTML dir=rtl>
  <Title>  حملة التكرار for</Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
    var x = 0;

    for( i=1; i<10 ; i++ ) {
           if( i % 2 == 0 ) continue;
           x += i;
    }
```

```
    alert("1 + 3 + 5 + 7 + 9 = " + x);
    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

ويكون الناتج كما يلي



جملة التكرار while

الهدف من جملة التكرار while هي تكرار مجموعة من الأكواد لعدد غير محدد يتوقف علي جملة شرط ، بدون وجود قيمة إبتدائية وبدون جملة للزيادة أو النقصان .

الصيغة العامة

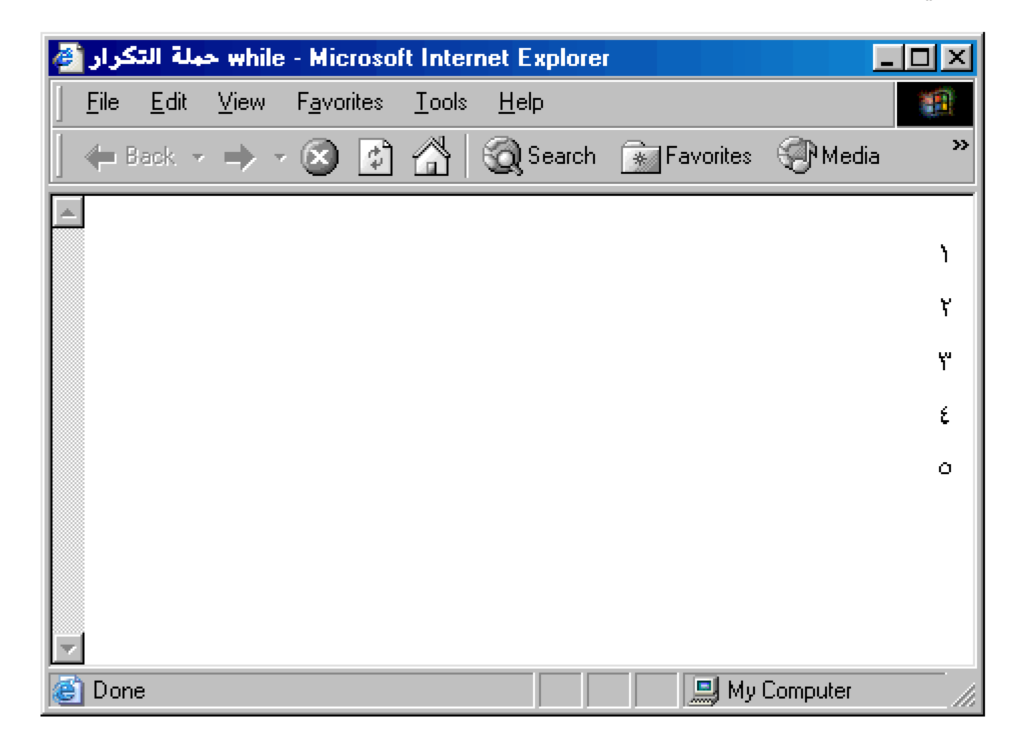
```
    while ( شرط التكرار ) {
           // الأكواد
    }
```

مثال
يقوم بطباعة الأرقام بين ١ و ٥

```
<HTML dir=rtl>
  <Title>  حملة التكرار while</Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
    var i = 1;

    while( i<6 ) {
           document.write( i + "<br>" );
           i++;
    }
    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

ويكون الناتج كما يلي :



من المثال السابق لا نجد فرق بين جملة التكرار while وجملة التكرار for
ولكن تظهر الحاجة إلي إستخدام جملة while عند التعرض إلي تنفيذ الكود بشكل متكرر حسب شرط غير مرتبط بعملية حسابية تكرارية كما بالمثال التالي :

مثال
نقوم في هذا المثال بالكشف عن كلمة السر
فلو أدخلت كلمة السر خطأ يتم إعادة إدخال كلمة السر حتي يتم إدخالها بشكل صحيح

```
<HTML dir=rtl>
  <Title> الكشف عن كلمة السر </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
           var userPassword = "123";

           var password = prompt("أدخل كلمة السر", "");

           while( password != userPassword ){
              alert( "كلمة السر غير صحيحة" );
              password = prompt("أدخل كلمة السر", "");
           }
```

```
        alert( "كلمة سر صحيحة \n مرحبا بك يا" );

    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

جملة التكرار do ... while

تتشابه مع وظيفة جملة التكرار while ، ولكنها تختلف في **أن الكود الموجود بداخلها يتم تنفيذه مرة واحدة علي الأقل** حتي ولو كانت جملة الشرط تساوي false عند أول مقارنة ،
لأنه يتم تنفيذ الكود أولا ثم يتم تنفيذ جملة الشرط .

الصيغة العامة

```
    do{
        // الأكواد
    }while ( شرط التكرار );
```

مثال
نقوم في هذا المثال بالكشف عن كلمة السر ، فلو أدخلت كلمة السر خطأ يتم إعادة إدخال كلمة السر وتتم هذه العملية ٣ مرات متتالية عند إدخال كلمة سر خاطئة .

```
<HTML dir=rtl>
  <Title> الكشف عن كلمة السر </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
        var userPassword = "123", password;
        var failureCount = 0;

        do{
            failureCount++;
            if( failureCount == 4 ) break;
            if( failureCount != 1 ) alert( "كلمة السرغير صحيحة" );

            password = prompt("أدخل كلمة السر", "");
        } while( password != userPassword );

        if( failureCount == 4 )
            alert( "غير مسموح لك بدخول تلك الصفحة" );
        else
            alert( "كلمة سر صحيحة \n مرحبا بك يا" + " : بعد عدد من المحاولات يساوي" + failureCount);
    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

أمثلة عامة عن جمل التكرار

مثال
حساب المتوسط الحسابي للأعداد من ٢٠ إلي ٨٠

```
<HTML dir=rtl>
  <Title> المتوسط الحسابي </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
           var count = 0, i;
           var sum = 0;

           for( i=20; i<=80 ; i++ ){
              sum += i;
              count++;
           }

           var average = sum / count;
           alert( "المتوسط الحسابي للأعداد من ٢٠ إلي ٨٠ يساوي : " + average );
    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

ويكون الناتج كما يلي :



مثال متقدم
نأخذ من المستخدم رقم بالنظام العشري ، ثم نحوله إلي النظام الثنائي مع عرض عدد البتات المكونة لهذا الرقم .

```
<HTML dir=rtl>
  <Title> التحويل للنظام الثنائي </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
           var decimalNumber, binaryNumber;

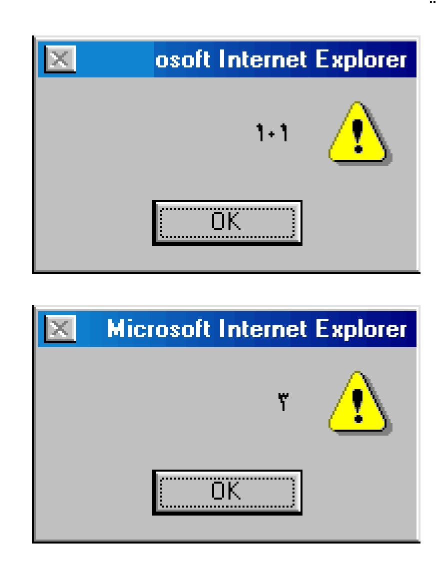
           decimalNumber = prompt("  أدخل الرقم العشري  ","");
```

```
            var binaryString = "", count=0;
            do{
               binaryString = (decimalNumber & 1) + binaryString;
               count++;
            } while( decimalNumber >>= 1 )

            alert(binaryString);
            alert(count)

    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

فعند إدخال الرقم ٥ يكون الناتج كما يلي



الفصل السادس

الدوال أو الوظائف

سوف نناقش إن شاء الله في هذا الفصل النقاط التالية :

- تمهيد
- تعريف الدوال أو الوظائف
- أنواع الدوال
- الصيغة العامة للدوال
- تعريف الدوال
- طريقة إستدعاء الدوال
- المتغيرات المحلية والعامة
- إسترجاع القيم من الدوال
- تمرير المعاملات للدوال
- إستدعاء الدالة لنفسها Recursive Function
- دالة داخل دالة inner Function

تمهيد

ربما كنا في الفصل السابق نحتاج إلي إستخدام جزء من الكود بشكل تكراري لذلك إحتجنا لجمل التكرار مثل جملة for ، ولكن لو كنا نريد إستخدام جزء من الكود بشكل دائما فسوف نلجأ إلي إستخدام ما يسمي **بالدوال أو الوظائف** Functions

لاحظ معي الحالة التالية

```
var password = prompt(" أدخل كلمة السر ","");

if ( password == "123" ){
     alert("مرحبا بك في موقعنا");
     document.write("مرحبا بك في موقعنا <br>");

     document.write("مدير الموقع");
}else if ( password == "111" ){
     alert("مرحبا بك في موقعنا");
     document.write("مرحبا بك في موقعنا <br>");

     document.write("مشترك بالموقع");
}else{
     alert("كلمة السر خاطئة");
}
```

لاحظ معي السطور التالية

```
     alert("مرحبا بك في موقعنا");
     document.write("مرحبا بك في موقعنا <br>");
```

فقد تم تكرارها مرتين ، فهب أننا سوف نستخدمها أكثر من مرتين فما الحل
هل نقوم بتكرار هذة الجمل كلما إحتجنا لها ، بالطبع هذا ليس بحل يروق لك
وأنا معك أن هذا الحل لا يروق لي أنا أيضا ،
لا تقلق الحل قد أتي بحمد الله فسوف نقوم بوضع هذيين السطرين داخل بوطقة (دالة أو وظيفة) ، نستطيع تنفيذ ما بها من الأكواد وقتما أردنا .

وتكون علي الشكل التالي

```
function displayMessage() {
     alert("مرحبا بك في موقعنا");
     document.write("مرحبا بك في موقعنا <br>");
}
```

وليس هناك اي داعي لمحاولة فهم ما كتب الأن ، ولكن تابع معي أول إنطلاقة حقيقية إلي عالم البرمجة مع الحديث عن الدوال أو الوظائف .

تعريف الدوال أو الوظائف

والوظائف هي مجموعة من الجمل يطلق عليها إسم ويشار إليها به وتنفذ كوحدة واحدة .

أنواع الدوال

يوجد نوعين من الدوال

- الدوال المبنية داخل لغة الجافا سكربت built-in functions
  هذه المجموعة من الدوال تم بنائها داخل لغة الجافا سكربت مثل دالة parseInt ، و يمكننا أستخدامها كأي دالة قمنا بتعريفها كما سنري فيما بعد .
- الدوال المبنية من خلال مبرمجي الجافا سكربت
  وهذا هو النوع من الدوال الذي سوف نتحدث عنه في هذا الفصل بشكل مفصل .

الصيغة العامة للدوال

```
function إسم الدالة ( معاملات الدالة ) {
     // الأكواد المراد تنفيذها

     // ربما تقوم الدالة بإرجاع قيمة
     return القيمة الراجعة ;
}
```

١- أولا نبدأ بكتابة كلمة function ثم يليها اسم الدالة و تخضع لنفس شروط تسمية المتغيرات ، بالإضافة إلي أنها لا يجب أن تأخذ اي اسم لدالة مبنية داخل لغة الجافا سكربت built-in function.

٢- ثم يتم تمرير معاملات لها أو ربما لا يتم تمرير معاملات لها كما سوف نري لاحقا .

٣- ثم يتم كتابة الأكواد المراد تنفيذها بين أقواس المجموعة الخاصة بالدالة .

٤- بعدها ربما تقوم هذة الدالة بإرجاع قيمة أو لا ترجع اي قيم كما سوف نري لاحقا .

تعريف الدوال

دعنا بعدما علمنا الصيغة العامة للدوال ، أن نقوم بكتابة احدي الدوال البسيطة وظيفتها عمل التالي أخذ اسم المستخدم ، ثم إظهار رسالة ترحيب له .

لذلك سنقوم بعمل دالة باسم showMessage هذه الدالة لا تأخذ معاملات ولا تقوم بإرجاع اي قيم كما يلي

```
function showMessage () {
     var userName = prompt("أدخل اسمك من فضلك","");
     alert(": مرحبا بك" + userName );
}
```

نعم لقد تمكنا من بناء هذه الدالة إستنادا علي ما تعلمناه للصيغة العامة للدوال
ولكن **كيف يمكننا تنفيذ ما بداخلها من أكواد ؟**
للإجابة علي هذا السؤال أتبعني في الفقرة التالية

طريقة إستدعاء الدوال

نعني بإستدعاء الدالة اي تنفيذ ما بداخلها من أكواد ، ويمكننا إستدعاء الدوال بعدة أشكال أحدهما :

- إستدعاء دالة من خارج اي دالة
- إستدعاء دالة من داخل دالة
- إستدعاء دالة لنفسها Recursive function

وهذا النوع من الدوال أكثر صعوبة ، سيتم نقاشه في أخر هذا الفصل

دعنا بعدما قمنا بتعريف الدالة showMessage أن نقوم بإستخدامها
كما بالمثال التالي

مثال

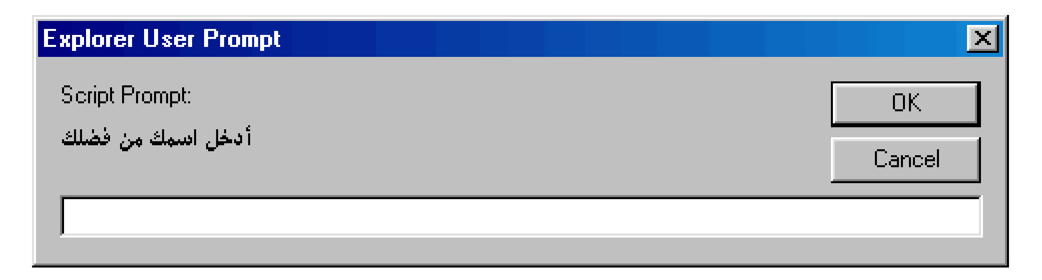
```
<HTML dir=rtl>
  <Title> نداء الدوال </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--

    // تعريف الدالة showMessage
    function showMessage () {
         var userName = prompt("أدخل اسمك من فضلك","");
         alert(": مرحبا بك" + userName );
    }

    // نقوم بإستدعاء الدالة showMessage
    showMessage();

    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

ويكون الناتج كالتالي



وبعد إدخال الاسم بالقيمة **محمد** يكون الناتج كالتالي



جميل جدا لقد تم تنفيذ أكواد هذا الدالة بمجرد كتابة اسمها متبوعا بأقواس كما بالسطر التالي

```
    // نقوم بإستدعاء الدالة showMessage
    showMessage();
```

لاحظ ما هو أت

ربما نحتاج في بعض الأحيان لعمل أكثر من دالة ، وربما نحتاج لتنفيذ أحدي الدوال من داخل دالة أخري كما بالمثال التالي

مثال

```
<HTML dir=rtl>
  <Title> نداء الدوال بعضها لبعض </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--
    var userName;

    // تعريف الدالة showMessage
    function showMessage () {
         // نقوم بإستدعاء الدالة getName
         // من داخل الدالة showMessage
         getName();
         alert(" : مرحبا بك" + userName );
    }
```

```
    // تعريف الدالة getName
    function getName () {
         userName = prompt("أدخل اسمك من فضلك","");
    }

    // نقوم بإستدعاء الدالة showMessage
    showMessage();

    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

ويكون الناتج كما بالمثال السابق

لاحظ معي السطور التالية
لقد تم إستدعاء الدالة getName من داخل الدالة showMessage

```
    // تعريف الدالة showMessage
    function showMessage () {
         // نقوم بإستدعاء الدالة getName
         // من داخل الدالة showMessage
         getName();
         alert(" : مرحبا بك" + userName );
    }

    // تعريف الدالة getName
    function getName () {
         userName = prompt("أدخل اسمك من فضلك","");
    }
```

لاحظ أيضا التالي
لقد تم تغير قيمة المتغير userName من داخل الدالة getName ، وتم عرض قيمته من داخل الدالة showMessage كما يلي

```
    var userName;

    // تعريف الدالة showMessage
    function showMessage () {
         // نقوم بإستدعاء الدالة getName
         // من داخل الدالة showMessage
         getName();
         alert(" : مرحبا بك" + userName );
    }

    // تعريف الدالة getName
    function getName () {
```

```
        userName = prompt("أدخل اسمك من فضلك","");
    }

    // نقوم بإستدعاء الدالة showMessage
    showMessage();
```

وهذا يقودنا إلي كيفية تعامل الدوال مع المتغيرات ، ولإكتشاف هذا دعنا ننتقل للفقرة التالية .

المتغيرات المحلية و العامة

المتغيرات المحلية Local variables

إذا تم تعريف المتغير داخل دالة ، فلن تستطيع عرض أو تعديل قيمة هذا المتغير إلا من داخل هذة الدالة المعرف بها ولا يمكنك عرض أو تعديل محتويات هذا المتغير من داخل دالة أخري ، ويسمي هذا المتغير **بمتغير محلي** Local variable كما بالمثال التالي

مثال

```
    // تعريف الدالة getJob
    function getJob () {
        var userJob = prompt("أدخل اسمك من فضلك","");
    }

    // تعريف الدالة showJob
    function showJob () {
        alert( " قيمة المتغير userJob = " + userJob );
    }
```

كما بالمثال السابق نجد أن المتغير المسمي userJob تم تعريفه داخل الدالة getJob ، لذلك يعتبر هذا المتغير متغير محلي بالنسبة للدالة getJob ولا يمكن التعامل مع المتغير userJob من خارج هذه الدالة أو من داخل دالة خارجية أخري.

ولو قمنا بعرض قيمة المتغير userJob من داخل الدالة showJob سيؤدي ذلك لحدوث خطأ كما بالسطر التالي

```
        alert( " قيمة المتغير userJob = " + userJob );
```

ملاحظة هامة

يمكنك إستخدام نفس اسم المتغير في أكثر من دالة بشرط أن يتم تعريفه داخل كل دالة علي حدي ، وسوف يتم التعامل مع كل متغير علي حدي داخل الدالة المعرف بها كما بالمثال التالي

مثال

```
<HTML dir=rtl>
  <Title> المتغيرات المحلية </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--

    // تعريف الدالة getJob
    function func1 () {
         var userName = prompt("أدخل اسمك من فضلك","");

         document.write( userName );
    }

    // تعريف الدالة showJob
    function func2 () {
         var userName = "محمد";

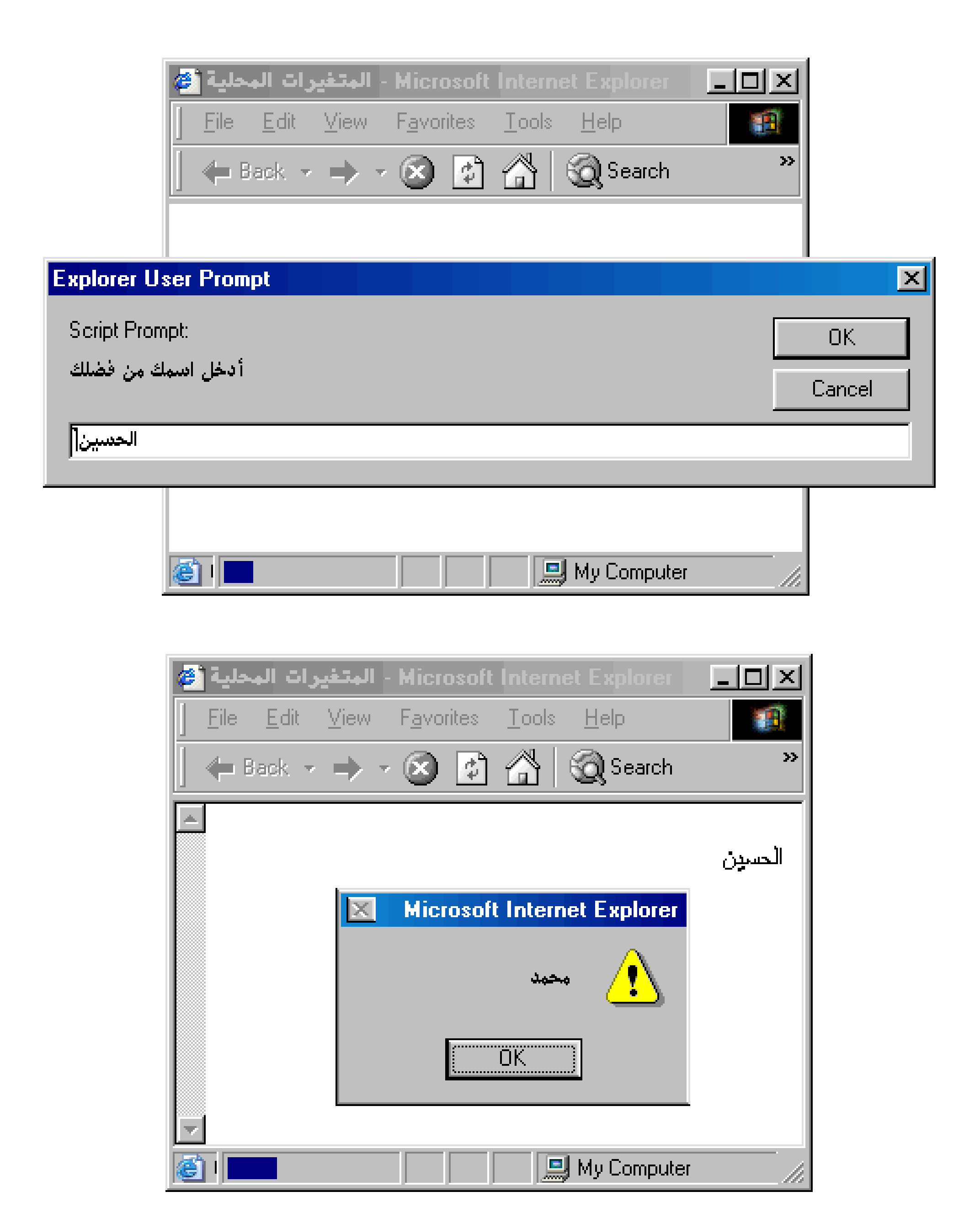
         alert( userName );
    }

    func1();
    func2();

    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

كما بالمثال السابق نجد أن المتغير المسمي userName تم تعريفه داخل الدالة func1 و الدالة func2 ، لكن يتم التعامل مع هذا المتغير علي أساس أنهم متغيرين كلا حسب مكان تعريفه داخل الدالة .

ويكون الناتج كما يلي



المتغيرات العامة Global variables

إذا تم تعريف المتغير خارج نطاق اي دالة اي علي مستوي الأسكربت العام ، بهذا تستطيع عرض أو تعديل قيمة هذا المتغير من داخل أي دالة أو من خارجهما ، ويسمي هذا المتغير **بمتغير عام** Global variable كما بالمثال التالي

مثال

```
var userName;

//  تعريف الدالة showMessage
function showMessage () {
     //  نقوم بإستدعاء الدالة getName
     //  من داخل الدالة showMessage
     getName();
     alert(" : مرحبا بك" + userName );
}

//  تعريف الدالة getName
function getName () {
     userName = prompt("أدخل اسمك من فضلك","");
}

//  نقوم بإستدعاء الدالة showMessage
showMessage();
```

كما بالمثال السابق نجد أن المتغير المسمي userName تم تعريفه في الكود خارج نطاق اي دالة ، وقد تم تغير قيمة المتغير userName من داخل الدالة getName ، وتم عرض قيمته من داخل الدالة showMessage

إسترجاع القيم من الدوال

يمكن للدوال في الجافا سكربت إرجاع قيم للجملة التي قامت بإستدعائها ، بإستخدام الكلمة المحجوزة return .
يتم وضع القيمة الراجعة من الدالة بعد الكلمة return .
القيمة الراجعة من الدالة يمكن تخزينها في متغير أو تكون وسط جملة حسابيه كما بالمثال التالي

مثال

```
<HTML dir=rtl>
  <Title> إسترجاع القيم من الدوال </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--

    //  تعريف الدالة getGuestName
    function getGuestName () {
         var userName = prompt("أدخل اسمك من فضلك","");
         return userName;
    }

     var returnName = getGuestName();
     alert( returnName );
    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

دعنا ندرس هذا المثال خطوة بخطوة
أولا قمنا بإستدعاء الدالة getGuestName
وبما أن هذه الدالة تقوم بإرجاع قيمة ، فسوف يتم تخزين القيمة الراجعة من الدالة في المتغير returnName
كما بالسطر التالي

```
     var returnName = getGuestName();
```

فعند إستدعاء الدالة يتم تنفيذ الأكود التالية

```
         var userName = prompt("أدخل اسمك من فضلك","");
         return userName;
```

وكما نري يتم تنفيذ الأمر prompt ويتم تخزين القيمة الراجعة منه في المتغير المسمي userName
ثم يتم إنهاء الدالة والخروج منها بتنفيذ الأمر التالي

```
         return userName;
```

ويقوم الأمر return بعمل شيئين أولهما **إرجاع القيمة التالية له** ، ثم **إنهاء تنفيذ الدالة** .

بعد إنتهاء تنفيذ الدالة ، يقوم مفسر اللغة interpreter بالإنتقال إلي مكان إستدعاء الدالة كما بالسطر التالي

```
     var returnName = getGuestName();
```

ثم يتم تخزين القيمة الراجعة من الدالة getGuestName في المتغير returnName .

ملاحظة
لاحظ أن اي أمر يكتب بعد الأمر return لا يتم تنفيذه ، لأننا كما أشرنا إلي أن بمجرد تنفيذ الأمر return يتم إرجاع القيمة التالية له ، ثم إنهاء الدالة

مثال

```
<HTML dir=rtl>
  <Title> إسترجاع القيم من الدوال </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--

    //  تعريف الدالة getGuestName
    function getGuestName () {
         var userName = prompt("أدخل اسمك من فضلك","");
         return userName;
         alert("إلي اللقاء");
    }

     var returnName = getGuestName();
     alert( returnName );
    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

فعند تشغيل هذا الكود لن تظهر رسالة **إلي اللقاء** لأنها أتت بعد الأمر return كما هو موضح

```
    //  تعريف الدالة getGuestName
    function getGuestName () {
         var userName = prompt("أدخل اسمك من فضلك","");
         return userName;
         alert("إلي اللقاء");
    }
```

تمرير المعاملات للدوال

معاملات الدالة :
هي قيم يتم تمريرها إلي الدالة من خارجها ، يتم إستخدامها داخل الدالة
ثم تمسح من ذاكرة البرنامج بمجرد إنتهاء عمل الدالة

طريقة تمرير المعاملات للدوال :
يتم تمرير المعاملات Arguments إلي الدوال بعدة طرق

- معامل واحد أو أكثر
- معاملات إختياريه Optional argument
- لا يتم عمل اي معاملات للداله ولكن يتم تمرير معاملات للداله بعدد غير محدد اي يمكنك تمرير اي عدد من القيم .

دوال ذات معامل أو أكثر

مثال

```
<HTML dir=rtl>
  <Title> تمرير القيم إلي الدوال </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--

     var GuestName;

    //  تعريف الدالة setGuestName
    function setGuestName ( name ) {
         GuestName = name;
    }

     setGuestName("الحسين");
     alert( GuestName );
    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

يتم تمرير المعاملات للدالة بين أقواس الدالة كما يلي

```
    //  تعريف الدالة setGuestName
    function setGuestName ( name ) {
```

ثم يتم إستخدام هذا المعامل علي انه متغير معرف داخل الدالة كما هم مبين بالسطر التالي

```
    function setGuestName ( name ) {
         GuestName = name;
```

ثم يتم إستدعاء الدالة كما يلي

```
    setGuestName( "الحسين" );
```

مثال
نريد عمل دالة يتم تمرير معاملين لها ثم تقوم بإرجاع القيمة الأكبر للقيمتين

```
<HTML dir=rtl>
  <Title> تمرير القيم إلي الدوال </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--

    // تعريف الدالة getMax
    function getMax ( num1 , num2 ) {
         return  ( num1 > num2 )? num1 : num2 ;
    }

     alert( getMax(21,34) );
    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

في الدالة getMax قمنا بتمرير معاملين كما يلي

```
    // تعريف الدالة getMax
    function getMax ( num1 , num2 ) {
```

ثم تقوم بإرجاع القيمة الأكبر للعددين الممرران لها كما يلي

```
    // تعريف الدالة getMax
    function getMax ( num1 , num2 ) {
         return  ( num1 > num2 )? num1 : num2 ;
    }
```

ثم يتم إستدعاء الدالة كما يلي

```
     alert( getMax(21,34) );
```

المعاملات الإختيارية Optional Arguments
نريد عمل دالة يتم تمرير معاملين لها ، ولكن المعامل الثاني يكون معامل إختياري optional argument
وظيفة هذه الدالة طباعة النص الممرر لها ، بحيث يكون المعامل الأول عبارة عن محتوي النص المراد طباعة ، أما المعامل الثاني يكون إختيار لإظهار رسالة ترحيب أم لا .

مثال

```
<HTML dir=rtl>
  <Title> تمرير القيم إلي الدوال </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--

    //  تعريف الدالة displayMessage
    function displayMessage ( messageText , dispalyHelloMessage ) {
         if( dispalyHelloMessage != null && dispalyHelloMessage )
             alert("مرحبا بك");

         alert( messageText );
    }

     displayMessage( "رسالة مرسلة" , true );
     displayMessage( "رسالة مرسلة" );

    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

في الدالة displayMessage قمنا بتمرير معاملين المعامل الثاني dispalyHelloMessage معامل إختياري اي ربما لا يمرر للدالة عند إستدعائها كما يلي

```
    //  تعريف الدالة displayMessage
    function displayMessage ( messageText , dispalyHelloMessage ) {
```

تم إستدعاء الدالة dispalyHelloMessage مرتين أحدهما تم تمرير المعامل الثاني لها بقيمة true والأخر لم يمرر لها إلا المعامل الأول فقط كما يلي

```
     displayMessage( "رسالة مرسلة" , true );
     displayMessage( "رسالة مرسلة" );

    //-->
    </SCRIPT>
```

تمرير عدد غير محدد من المعاملات إلي الدالة

فيها لا يتم عمل اي معاملات للداله ولكن يتم تمرير معاملات للداله بعدد غير محدد
اي يمكنك تمرير اي عدد من القيم .

مثال
نريد عمل دالة يمرر لها اي عدد من المعاملات ثم تقوم بإرجاع القيمة الأكبر لهذه القيم

```
<HTML dir=rtl>
  <Title> تمرير القيم إلي الدوال </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--

    // تعريف الدالة getMax
    function getMax () {
         var args = getMax.arguments;
         var max = args[0];

         for( var i=1; i<args.length; i++ ){
           if( max < args[i] )
               max = args[i];
         }

         return max;
    }

     alert( getMax(21,34) );
     alert( getMax(2,43,5) );
     alert( getMax(2,1,3,4,7,4,8,1) );

    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

وهذا المثال أقدمه لكم الأن بدون شرح حتي نتناول التعامل مع المصفوفات في الجزء الثالي من الكتاب

إستدعاء الدالة لنفسها Recursive Function

كما أشرنا سابقا أننا يمكننا إستدعاء دالة من داخل دالة أخري وقمنا بتطبيق أمثلة لذلك الغرض
ولكن هل يمكن لدالة أن تستدعي نفسها من داخلها ؟

أجيب بنعم ، ولكن بشرط وضع شرط ينهي إستدعاء الدالة لنفسها وإلا سوف يدخل البرنامج في حلقة من الدورات غير منتهيه infinite loop

مثال علي ذلك عمل دالة رياضية تقوم بحساب المضروب الرياضي Factorial لقيمة المعامل الممرر لها
يتم حساب المضروب الرياضي بالشكل التالي
مضروب ٣ يساوي ٣ x ٢ x ١
و مضروب ٤ يساوي ٤ x ٣ x ٢ x ١
و مضروب ٥ يساوي ٥ x ٤ x ٣ x ٢ x ١

مثال

```
<HTML dir=rtl>
  <Title> حساب المضروب </Title>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
    <!--

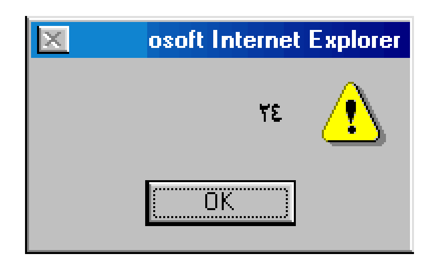
    //  تعريف الدالة factorial
    function factorial ( number ) {
           if( number == 1 || number == 0 ) return 1;

           return factorial( number-1 ) * number;
    }

     alert( factorial(1) );
     alert( factorial(4) );

    //-->
    </SCRIPT>
  </HEAD>
</HTML>
```

ويكون الناتج كما يلي





دالة داخل دالة inner Function

يمكننا تعريف دالة داخل دالة كما يلي

```
// تعريف الدالة superFunction
function superFunction ( number ) {

        // تعريف الدالة innerFunction
        // دالة داخلية
        function innerFunction ( number ) {

        }

        return "إلي اللقاء مع الجزء الثاني";
}
```

وهذا النوع المتقدم من الدوال سوف يتم شرحه في الجزء الثالث من هذا الكتاب الخاص بالبرمجة الكائنية للجافا سكربت .